Bin Yu

Data Compression For Energy-Efficiency Web Access On Mobile Devices

School of Electrical Engineering

Thesis submitted for examination for the degree of Master of Science in Technology.
Espoo 2.5.2013

Thesis supervisor:

Prof. Jukka Manner

Thesis advisor:

M.Sc. (Tech.) Le Wang

Aalto University
School of Electrical
Engineering

AALTO UNIVERSITY
SCHOOL OF ELECTRICAL ENGINEERING

ABSTRACT OF THE
MASTER'S THESIS

Author: Bin Yu

Title: Data Compression For Energy-Efficiency Web Access On Mobile Devices

Date: 2.5.2013 Language: English Number of pages:8+61

Department of Communications and Networking (Comnet)

Professorship: Networking Technology Code: S3029

Supervisor: Prof. Jukka Manner

Advisor: M.Sc. (Tech.) Le Wang

Nowadays, wireless data connections (2G, 3G and WiFi) have been the mainstream technologies for accessing Internet on modern mobile devices. However, users aware that heavy use of data transmission for web access via wireless interfaces leads battery life drain badly.

In order to extend battery life time and improve user experience, we present the solution for offering "energy-efficiency web access on mobile devices". A new compression strategy named selective-compression is introduced as an improvement of traditional HTTP compression in this thesis. The selective-compression strategy can properly handle binaries of web contents. And its mechanism relies on client/remote proxy pair structure.

From analysis of the experiment results, we make conclusion that the selective-compression strategy can bring nice benefits for energy saving and delay deduction on mobile devices while accessing web pages that include massive binaries. Furthermore, we give the suggestion to web developers and web service providers about how to create energy-efficient web pages.

Keywords: wireless communication; web browsing; energy efficiency; lossless compression; web proxy; selective compression strategy

Preface

This Master thesis has been written as a partial fulfillment for the Master of Science(Technology) degree at Aalto University School of Electrical Engineering, Finland. The work was conducted as a part of the "Future Internet" project in the Department of Communications and Networking at the Aalto University School of Electrical Engineering.

I would like to express my appreciation to all the people in our laboratory who supported and helped me in this thesis work. Especially, I would like to thank Professor Jukka Manner for offering me this valuable opportunity to work in this project under his guidance. In addition, I'm gratitude to my instructor Le Wang for his kindly support and guidance throughout the whole theoretical and practical research work.

I would like to owe my appreciation to one special people, Mr. Nässi Viktor. Many thanks for his kindly help with the hardware problems during the experiment phase of this thesis work.

Helsinki, 2.5.2013

Bin Yu

Contents

Symbols and abbreviations

Symbols

μJ	Microjoule
J	Joule
ms	Miliseconds
ECr	Energy consumption of data receiving over radio
ECd	Energy consumption of data decompression
DS	Data size in bits
ECrb	Energy consumption of receiving one bit data over radio
ECdb	Enery consumption of decompressing one bit data
V	Volt
P	Power
I	Current
U	Voltage
P_D	Power value of mobile device
U_P	Voltage provided by power supply
U_R	Voltage value of resistance
R_R	Resistance value of resistance
Mbits/s	Mega bits per second

Abbreviations

API	Application Programming Interface
BIN	Binary file
BMP	Bitmap
C	C programming language
C++	C plus plus programming language
EDGE	Enhanced Data Rates for GSM Evolution
EXE	Executable file
GIF	Graphics Interchange Format
GPRS	General Packet Radio Service
GPS	Global Positioning System
GSM	Global System for Mobile Communications
HSPA	High Speed Packet Access
HTTP	Hypertext Transfer Protocol
JPEG	Joint Photographic Experts Group
MP3	Moving Picture Experts Group Audio Layer III
NI	Network Interface
OBML	Opera Binary Markup Language
PC	Personal Computer
PDA	Personal Digital Assistant
PDF	Portable Document Format
PNG	Portable Network Graphics
QoE	Quality of Experience
RTT	Round-trip Delay Time
SWF	Shockwave Flash
UMTS	Universal Mobile Telecommunications System
URL	Uniform Resource Locator
WLAN	Wireless Local Area Network
WiFi	A synonym for "WLAN"
2G	Second Generation
3G	Third Generation

List of Figures

List of Tables

1 Introduction

In this chapter, we give the introduction of the research work. First we show the background of mobile technology deployment nowadays. Then we state high energy consumption problem during Internet access on mobile devices. After that we introduce existing solutions from earlier research works. At last we generally describe the contribution for improving battery life of mobile devices.

1.1 Background

During the past ten years, the development of Internet communication technology well followed Moore's Law and has improved tremendously. The trend of modern Internet communication is to connect modern devices with wireless technology. This was just treated as theoretical concept many years ago. However nowadays, the always-on, always available wireless Internet connection has become a reality in more and more areas of European countries.

There are many modern wireless connectivity technologies that can cover wide areas from slow to high data rates connection, such as, Global System for Mobile Communications (GSM)/General Packet Radio Service (GPRS), Enhanced Data Rates for GSM Evolution (EDGE), Universal Mobile Telecommunications System (UMTS) and High Speed Packet Access (HSPA). Also the Wireless Local Area Network (WLAN) hotspots, which is originally designed for stationary hosts to replace wiring network and to enable network roaming within certain area, can provide a high speed but more localized access for mobile terminals. These technologies were treated as high-end technologies few years ago. But now they are considered as commodity grade and are widely used in daily life. Furthermore, along with the fast development of hardware integration technology, the modern mobile terminals become portable, powerful and cheap. They are integrated with powerful computing processor(s) and massive storage memories. And they can carry advanced operating system platforms, which offer Personal Computer (PC) level user experience to mobile users and rich Application Programming Interfaces (API) to software de-

velopers. Furthermore, ordinary mobile phones that consumers can purchase with normal price in store are integrated with all standard mobile network modules. This greatly decreases the threshold of always-on mobile Internet access.

All of these improvements of modern technologies bring the mobile users a wealth of versatile services. Many cool and desktop-like applications, combining mobile functionality such as Global Positioning System (GPS) and telephony, appear on mobile terminals. One of the primary features on mobile devices is web browsing. The web browsers on modern mobile platforms can fetch wide variety of web contents from web servers through the mobile connection, including not only text contents, but also multimedia and other binary contents, for example, Flash files, Portable Document Format (PDF) files, and Joint Photographic Experts Group (JPEG) files. Then it can display these contents to mobile users. This feature has totally changed the user experience of mobile users to access Internet resources, such as searching at Google; and also the way how people connects and communicates with each other, for example, using Facebook and Twitter.

1.2 Problem Statement

Due to the fast development of mobile connectivity technology, the range of mobile communication is no longer restricted in local area (WLAN). Long distance mobile data connection becomes a normal scenario. Furthermore, the mobile service providers can offer mobile customers faste data link rates, in order to offer the customers best web surfing experience. After GSM/GPRS, EDGE, UMTS, and HSPA mobile network, recently, the 4G mobile connection technology becomes major trend, and brings much higher data link rate to customers. More and more mobile device manufactures start to produce products with 4G radio module integrated, in order to support this mobile ultra-broadband Internet access standard.

However, mobile devices are not passive devices that can operate without energy source. All the modern features of mobile device are driven by device's heart, that is, the battery. Unfortunately, the battery technology did not follow up the pace of the deployment of rich functionality on mobile terminals. Although the modern

batteries carried on mobile devices are becoming lighter, thinner and larger capacity, increased battery capacity is still like "a drop in the bucket" and far away from enough. In modern mobile devices, besides the screen, processor(s) and memory, the radio interfaces become major consumers of the limited battery energy. [1][2] Battery energy drains very fast not only while locally playing games or watching videos, but also while downloading Internet contents from web servers at high or low speed data link rate via radio interfaces. A common case is that user can only browse rich contents, for example, Moving Picture Experts Group Audio Layer III (MP3), images or Youtube videos, by using 3G service on mobile device in a few hours the best. User has to charge the device once per day after just short period use of mobile network. This problem greatly downgrades the user experience of mobile user. The so-called "always-on, always available" mobile Internet accessing is restricted by the battery problem. Thus, mobile manufactures now have to face this critical problem: along with the critical increasing of energy consumption by radio interfaces, the performance and life time of mobile devices are reaching bottleneck due to the lack of battery capability.

In fact, the radio interfaces on mobile device are big energy consumers. They use huge amount of energy while sending or receiving bytes. And even keeping radio interfaces on without using them still consumes considerable amount of energy. Nowadays, the web browsing is known as a major Internet service. Thus, under the premise of battery limitation, novel technologies are needed to reduce the radio energy consumption, in order to increase the battery life and improve use experience. In other words, it is so-called "energy-efficiency web browsing".

1.3 Related Solutions

Since the advent of mobile terminals, especially the mobile phones, many researchers have concerned their energy problem for many years. As mentioned before, on modern mobile phones, the wireless radio interfaces are big energy consumers. Thus, in some of early research works, the topics were focus on significantly decreasing energy consumption of wireless interfaces when transmitting data packages on mobile

devices. Fortunately, researchers have already proposed many solutions for this problem. These solutions are various sorts and implemented on different layers of mobile devices, for example, the hardware layer(radio interface), protocol layer(transport layer) and the application layer. Furthermore, some solutions achieve great results of energy efficient data transmission by the help of external devices, for example, a special proxy server on mobile core network. Anyhow, all of these solutions were proven to be more or less helpful for extending the battery life of mobile device. And some products, which implement energy efficient features, have been widely used for the wireless Internet data communication.

On hardware layer, the research work in paper [1] can be considered as one of the pioneers for measuring power consumption behavior of radio hardware interface. They assumed that it was possible for Personal Digital Assistant (PDA) to implement policies to control the sleep/active states of its network interface, in order to utilize its power resources efficiently without downgrading the user's ability to access information from mobile network. And they chose two Network Interfaces (NI), Metricon RicochetWireless Modem and the AT&T Wavelan, which were high-end devices at that moment but out of date nowadays, for their practical measurements. The measurement results were excited, and they clearly proved that the power consumption by network interfaces was a large partition of the total power usage by PDA. They found that the amount of time that network interface was in active state but without being used was the crucial reason of energy consumption. As the conclusion, their experiments explained that it was possible to achieve huge power saving by implementing power control strategy on network interfaces and using application specific policies.

In another research work [2], the researchers presented an interesting solution on protocol layer for energy saving purpose on mobile devices. They deeply studied and measured the energy consumption characteristics of three common mobile network technologies: 3G, GSM and WiFi. From the result, they clearly found that in GSM and 3G network there always existed a long period of high-power states, so called the "tail energy", after completing data transfer. This means that even we only transfer

one bit data through GSM or 3G wireless interface, constant energy consumption is still not avoidable. Therefore their target was to reduce energy consumption by using scheduling algorithm and prefetching technology to control the redundant energy tail of GSM and 3G network. They implemented the new protocol: "TailEnder" on mobile device. And based on their evaluation, the TailEnder brought great benefits to the email, news feeds and web search applications.

On application side, most existing solutions were invented for web browsing on mobile devices. Because of heavy energy consumption for even transferring one bit data through wireless interfaces, one approach for energy saving is to decrease the data size of web content. There are two choices to achieve this goal: compressing web contents or completely removing heavy contents from web pages. For compressing web contents before transmission, the Hypertext Transfer Protocol (HTTP) Compression [8] has been invented for many years. It is integrated as a common capability into the modern web client and web server, in order to decrease web content size and provide better data rate for data transmission between them. Normally, the HTTP Compression only compresses textual files, for example, .html and .css files, by using its data compressors: gzip and deflate [10]. Originally, it is designed for web browser on desktop operating systems. But nowadays, modern mobile web browsers also support it well. Furthermore, in order to reduce transmission data size by removing and rearranging the web content, some service companies have implemented flexible solutions by establishing performance enhancing and energy efficient proxy servers without being perceived by Internet content providers, for example, the Google's mobile internet proxy [11] and the Opera Mini [12]. Google's mobile internet proxy provides a browser-independent web page interface for mobile users to enter their target web site's Uniform Resource Locator (URL). Then the Google's proxy service fetches web contents, rearranges the alignment of web pages, and slims down the content sizes by removing large pictures and files. As the result, the mobile users will receive modified web page mainly consisting of textual contents and being improved for the small screen of mobile device in very short time. Similar, The Opera Mini is a web browser designed for mobile device. It integrates

the client proxy feature into the browser, and relies on the remote proxy to prefetch and reformat the original web page. Then it utilizes its own protocol, Opera Binary Markup Language (OBML) [13], to fetch processed data from proxy. Therefore only users of Opera Mini browser can enhance their browsing performance by using its remote proxy.

1.4 Our Contribution

As mentioned in previous chapter, one approach to save energy on mobile devices while browsing web pages, is downgrading the quality of web contents, in order to decrease the amount of data received via wireless interfaces. Most of mobile users are already used to good quality and full experience of web contents on the desktop environments. Thus reformatting web contents for mobile device will not be the goal of this thesis. Considering the limited processing capability and battery life on mobile devices, we would like to maintain the Quality of Experience (QoE) of web browsing as much as possible by using the data compression for mobile device.

There are two different sorts of compression methods: lossy data compression and lossless data compression. The lossy data compression can bring much better compression ratio for web content than lossless data compression. However, lossy data compression creates compressed data from original data with data loss. This will cut down the quality of data contents. So in this thesis, we only investigate benefits that lossless data compression can bring to mobile users. Actually, this topic is not new, and has been discussed in many early studies [3][4][5][6][9]. Also, the HTTP compression is an existing solution which has been included in HTTP standard as an standard option of HTTP header. It seems that nothing needs to be done anymore, and mobile users will be happy to receive longer battery life time by using HTTP compression. However, the mobile environment is different from desktop environment. Thus there are a few drawbacks in HTTP compression, which make it insufficient to be used for mobile web browsing. For example, the traditional HTTP compression doesn't provide any optimization facility for the limited battery life and processor capability on mobile devices. There also isn't any performance

measurement about using HTTP compression with different types of radio interfaces when it was designed. Due to these drawbacks of HTTP compression, the main interest and contribution of this thesis will focus on the following aspects:

- We measure and investigate the effect of different factors, such as wireless interface types and speed of data links, while using data compression on mobile devices, in order to provide more energy saving.
- We measure the benefit brought by compressing binary files, in order to figure out better energy-saving compression scheme than HTTP compression.
- We design and implement proxy pair architecture on both mobile client and remote proxy server, so that the new compression strategy will not affect the existing web servers on the Internet, and also decouple from specified web browser. The remote proxy only fetches web contents from web servers as normal. And the optimization of energy-efficient data compression will only happen between client and remote proxy pair. Then we introduce and analyze the benefits of new compression strategy under this proxy structure.

1.5 Thesis Structure

The rest chapters of this thesis are structured as follows: In Chapter 2, we will explain the theoretical knowledge about the energy consumption when transmitting data on mobile device via radio interfaces and the lossless data compression technology. And we will also extend the discussion about the HTTP compression and its limitations. Moreover, we will bring up the concept of the new compression strategy and prove it by mathematics formulas and logical deduction. In Chapter 3, we will explain how we implement the prototype applications, and how we measure the results of the experiments. In Chapter 4 we will show how we design three different experiments. Then in Chapter 5 we will analyze and discuss the measurement results of these experiments. Lastly in Chapter 6, we will make the conclusion about how much improvement we made in the new compression strategy, based on the

results of experiments. And also we will present the future works to improve the prototype of the new compression strategy.

2 Using Data Compression For Web Accessing

This chapter contains two sections. First, we present detailed knowledge about data compression theory on mobile device. Second, we explain the meaning of the energy-aware compression for mobile device in this research.

2.1 Theoretical Basis of Data Compression

The data compression is the core technology for achieving energy saving on mobile devices in the research work. Therefore in this chapter, we introduce the lossless data compression, and the advantages of using lossless data compression for energy saving on mobile devices. After that we point out the limitation of using traditional HTTP compression on mobile devices.

2.1.1 Lossless Data Compressions

The data compression methods can be grouped into two categories: lossy and lossless compression. All of these methods can achieve certain amount of data compression ratio. As the name suggests, lossy compression methods cause data loss during compression operation. It is acceptable for mobile users to degrade the quality of some types of data, such as audio and video. However, many other data types (e.g. textual data) must be transmitted faithfully without any information loss. As we discussed in Chapter 1.4, we cleared that our goal is to reduce energy consumption without downgrading user experience. Thus in this chapter, the lossless data compression is exactly the topic that we are discussing.

In computer science world, the lossless compression is an approach to pack identical information of original data into compression format without any data loss. [14] That means we can reconstruct the compressed data identical to original data by lossless decompression. Further discussion about the theory of lossless compression is out of the range of this thesis. What we concern is the performance of lossless compression methods on the aspect of energy efficiency on mobile device. There are many existing lossless compression methods, but not all of them can meet

the requirement as a compressor to offer high compression ratio and fast compression/decompression speed. After carefully selection, we decided to measure four well-known lossless compression tools and further discuss their benefits on mobile device in Chapter 3. These compression tools are, zlib, lzo, lzma and bzip2.

The zlib [15] is a well-known lossless compression library, which combines LZ77 and Huffman coding to form algorithm known as "deflate". These algorithms give zlib good compression on different variety of data with small system resource usage. Also, the popular gzip utility uses zlib as core compressor, and it is defined as one of the standard compressors in HTTP compression. There is not any limitation to the length of data that can be compressed or decompressed in zlib. And it provides nine different levels to control and balance its memory usage and compression speed. Higher compression level means faster but more memory consumption and lower compression level means slow but less memory consumption. The level facility of zlib gives developers choices to select a real energy-efficient level while compressing data on mobile device.

The LZO [16] is another good compression library meant for "real-time" compression. Like zlib, LZO uses LZ77 algorithm, and it has an additional hash table to perform searching operation. The special point of LZO compared with zlib is that it has asymmetric characteristic between compression and decompression. The compression speed of LZO is comparable with zlib. And the decompression speed is even faster. It has compression level facility like zlib, and these levels affect the size of additional buffer during compression operation. It doesn't require more memory for decompression. Furthermore, developer can use compression levels to trade-off between compression ratio and compression speed during compression operation. However it doesn't affect the decompression speed no matter which level is used for compression.

The LZMA [17] is a famous compression tool, which uses improved version of LZ77 algorithm namely Lempel-Ziv-Markovchain-Algorithm for data compression. This algorithm combines advantages of traditional LZ77 and deflate algorithms, so that it can achieve better compression performance. Due to its good compres-

sion/decompression speed and multi-threading support, the lzma has great potential to achieve good compression ratio for compressing any type of file, and to obtain energy saving on mobile devices.

The bzip2 [18] is a more efficient compressor than zlib but with lower compression speed. It compresses data in blocks, and it applies the Burrows Wheeler Transform (BWT) algorithm to represent frequently-recurring character sequences into compression format. Like LZO, the performance of compression and decompression of bzip2 is asymmetrics, because that decompression is faster. Developers have less control to the performance and memory consumption of bzip2 because there isn't level facility in it.

2.1.2 Advantages of Using Data Compression

Perhaps you may have a question in mind: why would the compression be an approach of energy efficient web accessing on mobile device? As we mentioned in Chapter 1.3, the energy-saving on mobile device has been an old research topic for many years. Many researchers have already figured out many different approaches [1][2][4][5][6][7][9] to achieve energy-efficient on mobile device. They did sorts of measurements of energy consumption on mobile devices in different ways. However, almost all of their research works got similar result: it is quite heavy to transmit even one bit data via modern radio interfaces; and decreasing transmission data is valuable for extending battery life on mobile devices.

As a common knowledge, we know that the data compression is to decrease usage of storage resource of files by encoding information into less bits than its original size. Therefore on mobile device, using compression allows us to transmit same information in less bytes, in order to bring higher information bit rate on radio interfaces. The compression encoding task relies on mathematics algorithm and the computation capability of computer processor(s). If we measure the energy consumed on a single bit transmission over wireless, generally it consumes energy over 1000 times greater than a single 32-bit CPU computation.[3] Thus, theoretically if we can use 1000 computation operations to compress/decompress data by even

one bit, energy would be saved for sending/receiving this data.

Another benefit brought by compression is lower latency of data transmission. This is mainly because of the decreasing of the amount of data content. As mentioned before, the radio interfaces consume considerable energy while they are active. Based on previous measurement mentioned in [7], the transmission Round-trip delay Time (RTT) intensively affects energy efficiency. The energy consumption per bit significantly increases from 0.536 uJ/bit upto 2.103 uJ/bit, when RTT rises from 60ms to 1060ms. Thus if we can shorten the total time of data transmission, we will get chance to save energy. This effect is not obvious on fast radio interfaces, such as 3G or WiFi. However, the low speed link like 2G/2.5G can gain much benefit from decreasing transmission time by spending time on compressing data.

These are important proofs for supporting the benefit of energy-saving by data compression. However, there are massive types of compression algorithms that can be chosen, and they have different behaviors from energy-efficient aspect, which have been measured in paper [6]. Thus we need to carefully figure out which compression methods are suitable to achieve energy saving. Moreover, another showstopper also affects the way we select the compression method, that is, energy consumption per bit on radio interfaces. This value is various among different types of radio interfaces, for example, 2G/2.5G, 3G and WiFi. Thus in order to maximum the benefit brought by data compression, we need to consider these two factors, and figure out exactly in what kind of situation we can use data compression. We will discuss this deeply in Chapter 2.2 later.

2.1.3 HTTP Compression and Its Limitations

In Chapter 1.3, we referred HTTP compression as an existing technology for web content compression. The HTTP compression is a capability that was designed for HTTP standard in order to better use network bandwidth and speed up the transmission between web browser and server. The gzip and deflate are included as compressors of HTTP compression to provide nice compression performance. [8] Although the HTTP compression was invented many years ago, it is still widely

used in desktop environment nowadays. However, in mobile world, the HTTP compression is more or less obsolete. This is because that the benchmark for deciding good compressors on mobile device is different. Many new factors, such as hardware limitation and radio link status, will affect the selection of compressors. These factors must be taken into consider by developers in order to achieve good compression performance on mobile system. In view of these factors, the drawbacks of old HTTP compression have been revealed:

- First, the HTTP compression was originally designed for desktop browser. As we know, on desktop computer there is plenty of processing capability and energy to use, thus the web server can use any compressor of HTTP compression to achieve best compression ratios for web contents when sending them to browser. And then the browser can decompress them at highest speed without any worry about energy consumption. However on mobile device it is another story. Because of the restriction of battery life and processing capability, decompressing compressed data also consumes energy. Thus we need to carefully select a proper compressor, in order to balance the trade-off between the energy spent by decompressing data and the energy saved from data size deduction by data compression. There are many factors that will affect the choice of compressor, for example, the type of wireless interface in use and the current link date rate. Traditional HTTP compression doesn't handle any of these factors well for mobile device when it was invented. It only uses limited number of compressors, such as deflate and gzip, for all kind of files. However, sometimes as mentioned in Chapter 2.1.1, different compressors can bring different compression ratios to same file.

- Second, another limitation of HTTP compression on mobile device is that it only compresses textual files of web contents, including .html files, .css files, .js files, .txt files and so on. We have to say that this is a wise solution on desktop environment. The modern web contents mainly consists of two sorts of files: textual and binary files. We know that binary files, such as image

files or music files, have large data information originally. In order to decrease the content size of web page and show better browsing experience to users, the web page providers make most of binary files into compression format, for example, .jpeg for image and .mp3 for music. Thus, these files have little space to be compressed any more. And only textual files can get nice compression ratios. However, in mobile environment, it is great worthy to decrease even one bit date transmission to save energy, because of heavy energy consumed by transferring data via wireless interfaces. Thus, besides compressing textual files, in special situation compressing binary files can still bring energy-saving, even it is not possible to get better compression ratios for these files.

- Last but not least, HTTP compression has been supported by many browsers and web servers for many years. However, still not all web servers deploy this feature. Maybe the server administrator forgot to enable it, or maybe they thought it is not crucial to use it. Anyhow, this is tricky but true that we cannot force all the web providers to offer this nice feature for us.

Nothing is perfect, so is HTTP compression. However, because of its imperfect, we are driven to deeply measure and investigate the behaviors of different compressors on mobile device, and figure out the new road to achieve the energy-aware compression.

2.2 Energy-Aware Compression For Mobile Device

In this chapter, we use mathematical formulas to prove how to achieve energy-aware compression on mobile device. And then we define new categories for web contents. At last, we explain how the new energy-aware compression strategy should be implemented in real environment.

2.2.1 Formulas of Energy-Saving Concept

In Chapter 2.1, we introduced lossless data compression and the possibility of saving energy by using data compression for mobile data transmission. However, it is not

silver bullet. The data compression introduces considerable reduction of content size. Therefore, it should effectively decrease energy consumption of radio interfaces as it is supposed to. However, the trade-off is the energy usage of the intensive computation and memory access for compression/decompression operation. For this reason, the consequence brought by using data compression on mobile devices may be counterproductive in some situations. For example, sometimes when we receive compressed data on mobile device, the energy consumption of decompressing data is more than the energy saved by data compression. Then it doesn't make any sense to implement data compression on such data. The main reason that causes this phenomenon is that the energy consumption of receiving and decompressing one bit data are not constant values. We know that there are several factors that can affect those values on real-time, such as wireless link rate (2G, 3G and WLAN), transmitted data type (easy-to-compress and hard-to-compress) and mobile hardware type (CPU and RAM). [7]

These factors cannot be controlled by us because that different user can use different mobile device and can transmit different types of data via different wireless data links. Therefore, in this chapter, our task is to derive a formula which can be used real-timely based on different combination of factors, in order to figure out on which condition it is possible to save energy by using data compression during wireless data transmission.

There are two actions of data transmission: data sending and data receiving. While browsing web pages on mobile devices, the receiving action is the most common action. Therefore, here we only take receiving action on the mobile receiver side as example. We assume a situation that sender sends compressed data to mobile receiver, and the receiver decompresses the data after receiving the compressed data. Let us assume that ECr stands for "energy consumption of data receiving over radio", and ECd stands for "energy consumption of data decompression". Then simplifying we can formalize the original formula of energy saving by compression during receiving data as follow:

$$ECr(compresseddata) + ECd(rawdata) \leq ECr(rawdata) \quad (1)$$

The formula above means that the energy consumption of receiving data can be saved when the total energy consumption of receiving compressed data and energy consumption of decompressing this compressed data into raw data is less than the energy consumption of receiving uncompressed raw data.

However, this formula is quite general and we need to break it down into details. We can assume that DS stands for "data size in bits", ECrb stands for "energy consumption of receiving one bit data over radio", and ECdb stands for "enery consumption of decompressing one bit data". Then we can derive the formula into:

$$DS(compresseddata) \times ECrb + DS(rawdata) \times ECdb \leq DS(rawdata) \times ECrb \quad (2)$$

Furthermore, because the original size of raw data minus compressed size of raw data is the reduced data size by compression, then we can simplify it as:

$$DS(rawdata) \times ECdb \leq DS(reduceddata) \times ECrb \quad (3)$$

Now we are closing the goal. The ratio $\frac{DS(reduceddata)}{DS(rawdata)}$ here is called "compression ratio". And the ECdb and ECrb is different between mobile devices on different wireless links. However the ECdb is approximately constant and the ECrb can be measured on specific mobile device with different wireless data links (2G, 3G and WLAN). Finally, we get the final fomula:

$$DataCompressionRatio \geq \frac{ECdb}{ECrb} \quad (4)$$

This formula explains the proper condition to implement data compression for energy saving, that is, when the data compression ratio is greater than the value of $\frac{ECdb}{ECrb}$. As we mentioned before, this value can be measured on specific mobile device in specific situation. We have already done this measurement in the previous work

[7] on Nokia N900 mobile phone [20] with 2G, 3G and WLAN wireless networks. And the Table 1 lists the energy-efficient data compress ratios on different wireless links.

Table 1: Energy-efficient compression ratios on different wireless links

	2G	3G	WLAN
Ratio	0.2%	1.5%	4.0%

Therefore, the energy-saving solution comes to be simple enough. Before transmitting data to mobile receiver, the sender side can compress every data file and examine its data compression ratio. If the compression ratio of this file is greater than the minimum ratio of specific wireless data link that the mobile device is using, then the sender could send the compressed data for energy-saving. Otherwise, the raw data should be sent instead.

2.2.2 New Categories of Web Content Files

Nowadays, modern web pages contain many different types of rich web contents, besides the original web contents, such as text files and images. Users can enjoy more interesting web contents via modern web browsers, like musics, animations and videos. However, these rich contents bring more challenge to the energy saving on mobile devices. The battery drains very fast when user is listening music or watching video via wireless data links. As we discussed in Chapter 2.2.1, we figured out a new approach to achieve energy-efficient by using data compression when receiving data on mobile devices via radio interfaces. Therefore, in this chapter, we will refresh the understanding of the characteristics of web page contents and then re-classify them into new groups, in order to support the new compression approach.

Generally in most web pages, the textual files consist of the major part as their web contents, including html, css and script files. Therefore, usually people classifies web contents into two major types: textual contents and non-textual contents which are also called "binaries", such as JPEG, MP3 and Shockwave Flash (SWF) files.

As we mentioned in Chapter 2.1.3, the traditional HTTP compression follows this classification way to select compression target. It only compresses textual files by default due to the good compression ratios of these files. Obviously this leads the limitation of using HTTP compression on mobile devices. And a new way of data type classification is needed. According to the analysis in previous research work [16], we found another way from the perspective of compression ratio to classify web contents. Then those contents can be classified into three new categories, namely easy-to-compress files, compressible files and hard-to-compress files. The easy-to-compress files mostly include textual files and some other binaries, such as Binary file (BIN) files and Bitmap (BMP) image files. These files can achieve great compression ratio by using almost any compressor. Therefore we can always obtain energy-saving by compressing these files. Some binaries, for example, JPEG image files, MP3 music files and Executable file (EXE) executable files were designed originally to encrypt massive information into small size with some data losing which is acceptable for user experience, in order to be suitable for being transmitted or be shared on the Internet. These files are usually in compressed format. And sometimes the result of compressing those files will be counterproductive, which means that compression cannot help with decreasing their sizes. Therefore we call these files as hard-to-compress files. Last but not least, some special web contents namely compressible files are more and more shared through Internet, such as the PDF files and the SWF files. We treat them as special file types, because that they are neither pure textual files, nor fully compressed files as well. Their special characteristics were revealed in this research that based on specific data content what they contain and by using specific compressor, these files can still achieve better compression ratios than aforementioned hard-to-compress files. [16] In traditional HTTP compression these files are ignored when applying data compression. However, in order to save energy as much as possible on mobile device, they will be carefully treated in the new compression scheme.

As mentioned before, our target is energy-saving on mobile device. And energy consumption on single bit transmitted over wireless link is costly, especially over slow

links. Consequently, the new classification method of web contents is meaningful, because it reveals us the fact that the energy-saving by compressing non-textual files can also be considerable. And it brings us the possibility of implementing selective-compression to web contents on web proxy/server side before transmitting them to mobile devices, in order to achieve energy-efficient data receiving on mobile devices.

2.2.3 Selective-Compressing Web Contents

In previous chapter, we explained the new approach to classify web contents. And furthermore, we brought forward brief introduction of the new compression scheme at the end of previous chapter. In this chapter, we will introduce more details about how the selective-compression strategy works in this research work.

Our main research goal is to save energy on mobile side which only has limited battery life by delegating the task of fetching and select-compressing web content to web server/proxy, which has stable power supplier and powerful computation capability. According to the formula we derived in Chapter 2.2.1, we can select web contents that need to be compressed before forwarding them to mobile receivers based on the their compression ratios. And from Chapter 2.1.2 we know that there are a large amount of "binaries" which are still "compressible" and could benefit the energy-saving on receiver side. Therefore, the textual files are still handled as same as HTTP compression in our solution. Unlike the HTTP compression, the selective-compression scheme tries to fully take advantage of powerful resources of web server/proxy by pre-selecting and compressing compressible binary files before transmission. This compression decision is only made when the condition fulfills the energy-efficient selection formula. The server/proxy first tries to compress all binaries of web contents and calculates their compression ratios. Because the server/proxy can be notified what wireless data link the mobile device is using, it can then apply the proper threshold value of energy-efficient compression ratio and compare the ratio with all compression ratios of binaries. The binaries which have better compression ratios than the threshold can be substituted by their compressed files. And the ones cannot be compressed anymore will be kept for sending. Finally,

after finishing these selection operations, new web contents including compressed and uncompressed contents will be forwarded to mobile device as usual.

Someone may doubt this solution because that despite the selective-compression strategy can decrease the data content with energy-efficient, however it may bring deduction of user experience of web browsing on mobile device. For example, if certain web page contains huge amount of binaries, then the web server/proxy have to compress and check all of them, and additional delay may happen. Furthermore, because the mobile devices receive many compressed files, the decompression operation will be also heavy because of the limited computation ability of mobile devices, and spend more time than before. However, we cannot underestimate both of the capability of modern web server/proxy and the benefit that data compression brings. On server/proxy side, nowadays the computing capability of web server/proxy is more powerful than years before. Therefore compressing several binaries only takes few seconds which cannot be aware by mobile users. On mobile device side, although decompressing data takes time, the receiving time decreased by data compression is also considerable. Especially while using low date rate wireless links for receiving data, the data compression can decrease huge receiving delay. Therefore, the data compression will not increase delay of web browsing. And it will speed up web browsing significantly in specific situation.

Until here, we have approved the feasibility of energy-efficient selective-compression to web contents from the theoretical aspect and mathematics aspect. Everything is in order except what is crucial, that is, implementing this new solution for practical experiments. Therefore, we will focus on this task in next chapters.

2.3 Summary

In the chapter 2, we first introduced the concept of lossless data compression, and the theoretical advantages of using lossless data compression on mobile device to save energy and decrease transmission delay. In normal cases, using HTTP compression can more or less gain energy-efficient benefits. However, because of the limitations of the traditional HTTP compression, we tried to figure out a better solution. There-

fore, we proposed a new compression strategy to bring more energy-efficient benefits by data compression on mobile devices. In order to prove the feasibility of the new strategy in theory, we deduced the energy-efficient compression ratios as the baseline of achieving energy-saving by using data compression. And we reclassified the web contents into different groups based on the their compressibilities. At the end, we designed the new selective-compression strategy based on our theoretical findings. And the new selective-compression strategy leaded us to implement the prototype of the client/remote proxy pair for the practical experiments.

3 Implementation and Measurement Setup

This chapter includes two sections. First, we illustrate the implementation details about our test proxy-pair on which the selective-compression strategy was implemented. Second, we present how to setup the measurement environment for supporting our experiments, and discuss the measurement theory used in these experiments

3.1 Application Implementation

In order to measure the real efficiency of the new selective-compression strategy in real environment, we designed and implemented proxy prototypes on both mobile client and remote proxy. In this chapter, we will present the detailed operational theory of our client and remote proxy prototypes.

3.1.1 Proxy Components

Before introducing the operation logic of client and remote proxy, we need to first understand these component libraries which consist of the core parts in both client and remote proxies. These libraries include libcurl, libmicrohttpd, zlib and lzo. We have already introduced zlib and lzo compression libraries in previous chapter. Therefore, here we will focus on the details about libcurl and libmocrohttpd.

The libcurl [21] acts as the core library of curl which is a wide-used command line tool for transferring data with URL syntax over Internet. The libcurl supports various Internet protocols such as HTTP, FTP, and so on. And it is a free and open-source library. It is written in C language. And it is highly portable and supports many different operating systems, such as Windows, Linux and Mac Os. Because of its opening and powerful features, many applications use it for Internet data transfer, for example, the famous distributed version control system "git". We use libcurl as the "web client" in our proxies. This means that the libcurl is mainly used to request and fetch web contents from Internet.

Another core component of our proxies is the libmicrohttpd [22]. If we treat libcurl as the "web client" part in our proxy, then the libmicrohttpd is the "web

server" part. We choose it because that the libmicrohttpd is open-source and lightweight HTTP server library which is written in C language. It has nice run-time efficiency and small size. And it has similar functionality as Apache HTTP server. Therefore, it can embed its HTTP server features easily into other applications. The libmicrohttpd also has very simple and expressive API. The implementation of its API is HTTP 1.1 compliant. It can support multiple thread models, which is very useful to improve the processing efficiency of server. Therefore, in our proxies the libmicrohttpd is used to listen and response the HTTP requests from the web browser and libcurl.

3.1.2 Client Proxy

Now we can start to explain the structure of client proxy. Just as the name implies, the client proxy is the proxy on the mobile client side of our proxy pair. This proxy runs on the background process of mobile device, and acts as a black box for any mobile web browser. Generally, the mobile web browser treats client proxy as a normal local proxy, and sends HTTP requests to it. The client proxy catches these requests and forwards them to the remote proxy. When the client proxy receives the response content from remote proxy, it will first handle the received data and modify it into uncompressed format. Finally the client proxy sends the raw data back to mobile web browser. All of these steps seem to be transparent to mobile web browser. And the web browser does not notice any data processing on the web contents that it requested.

The Figure 1 shows us how the inner structure of our client proxy looks. The green arrows stand for the in and out data flows of the client proxy. From outside view of client proxy, the data flows seem to be symmetrical. However, inside the proxy, the data flows are asymmetrical. The red arrows indicate the incoming data flow from the mobile web browser. Typically this data flow is always HTTP request for web page content. The web server module (libmicrohttpd) receives this request and passes it to the web client module (libcurl) without any change. Then the web client module sends the request out to the remote proxy. Until here, the process is all

the same as the normal HTTP request from web browser. However, we should also notice that the data from remote proxy may be in compression format. Therefore, the blue arrows explain the path of incoming data flow from remote proxy. When the web client module gets response from remote proxy, it will first pass the data to decompression module. The decompression module checks whether the data is compressed or not. And if it is compressed, then decompression module checks which compression algorithm the data is applied. After the checking and processing by decompression module, the normal web content will be sent back to mobile web browser for displaying to user.

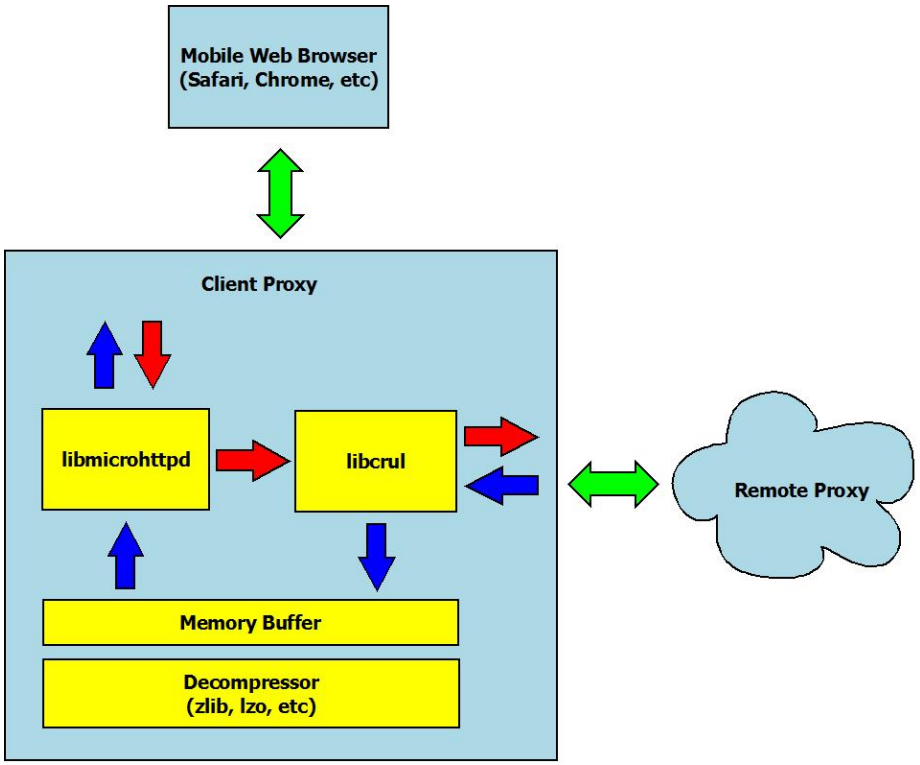

Figure 1: Structure of Client Proxy

3.1.3 Remote Proxy

After explaining how the client proxy works, then we can reveal the structure of remote proxy. From the Figure 2 we can notice that its structure is similar to the client proxy in many major places. It consists of three main components as well. And the web client and web server components have the same feature as those in the

client proxy. There are two main differences between the client proxy and the remote proxy. First, the remote proxy applies compression module instead of decompression module. And furthermore, it implements the selective-compression logic which relies on the energy-efficiency compression ratios. Therefore, The compression module is the significant component of remote proxy. Second, instead of communicating with another proxy, the web client module of remote proxy exchanges HTTP messages with real web server, in order to prevent web servers from noticing the existence of this proxy server. The web server will treat our remote proxy as normal HTTP client, and response the requested web contents back to remote proxy as usual.

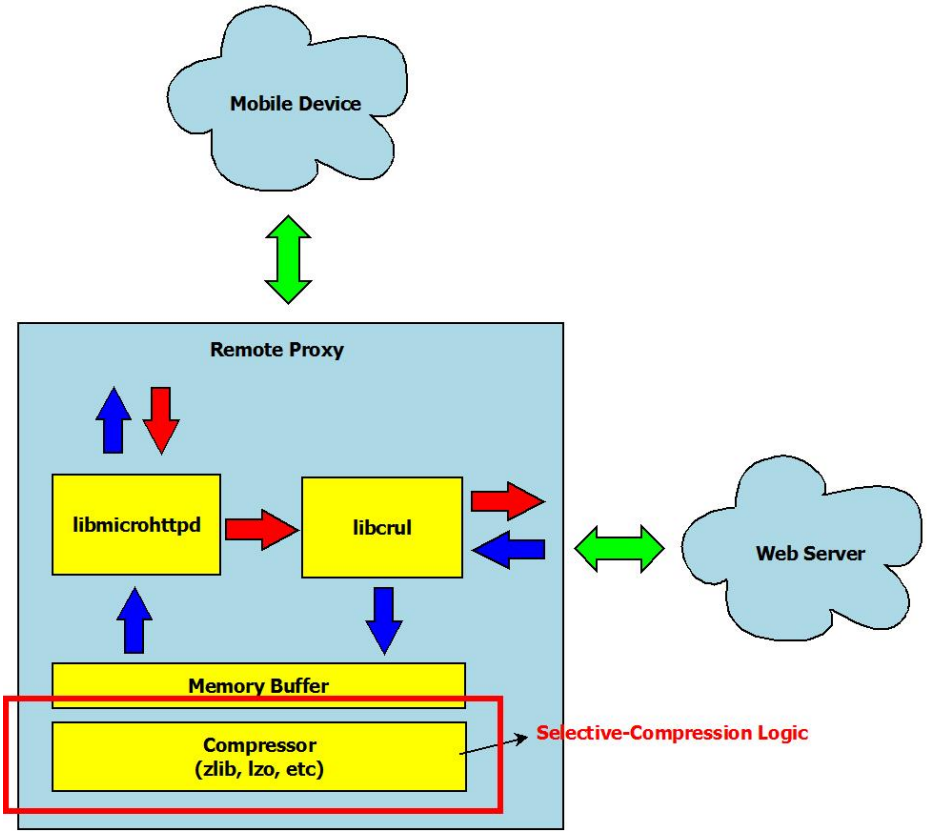

Figure 2: Structure of Remote Proxy

3.2 Measurement Setup

Understanding the measurement environment for our experiments is important before we introduce the design of experiments. Therefore, in this chapter, we will

present the measurement devices and explain the detailed theory for measuring useful data in our experiments.

3.2.1 Measurement Devices

Before we start to introduce each experiment design, let us first get familiar with the devices in each experiment. We choose the Nokia N-Series mobile devices, N810 Internet tablet and N900 smart phone, as the mobile devices in our experiments. And We adopt the NI cRIO-9215 analog input module to measure the power fluctuation on mobile devices.

Nokia N810 Internet Tablet

Figure 3: Nokia N810

The Nokia N810 Internet tablet [19] is one of the great classic Internet mobile device which was announced by Nokia in 2007. It is recognized as a pioneer and representative of modern mobile device. The hardware configuration of this device has many highlights at that moment:

- Chipset: TI OMAP 2420 (Open Multimedia Application Platform)
- CPU: 400MHz, built on ARM 1136 with 32KB data cache and 32 KB instruction cache

- Memory: 128 MB RAM and 256MB ROM; 2 GB storage
- OS: Maemo Linux OS
- Wireless interfaces: IEEE WiFi 802.11b/g
- Battery: Li-lon 1500 mAh battery
- Browser: xHTML, HTML, Adobe Flash

From the hardware specification of N810, it is clear that this portable device has almost full features the same as a laptop computer. It has acceptable processor and enough memory space for processing compression algorithms. And it supports standard WiFi communication. Furthermore, the highlight is that it runs Maemo [23] system, which is an open-source operating system based on Linux kernel. Therefore, we can develop and deploy our proxy binaries written in C/C++ easily to this device without much code-portable issue. And we can use many handy open-source tools to control the hardware actions and performance of N810, for example, the link rate of its wireless interface. All these advantages lead us to choose this device in our experiments.

Nokia N900 Smart Phone

Figure 4: Nokia N900

About two years later after the announcement of N810, Nokia released its first smart phone, Nokia N900 [20]. It inherited advantages of N810's design, and evolved its hardware in order to catch up the mainstream configuration at that time. Its hardware specification is:

- Chipset: TI OMAP 3430
- CPU: 600MHz ARM Cortex-A8
- Memory: 256MB RAM; 32 GB storage
- OS: Maemo 5 Linux OS
- Wireless interfaces: IEEE WiFi 802.11b/g, quad-band GSM, UMTS/HSDPA
- Battery: Li-lon 1320 mAh battery
- Browser: xHTML, HTML, Adobe Flash

Comparing with N810, the hardware of Nokia N900 is more powerful. It has faster processor and larger memory space, which can be used to handle data compression better. Besides the standard WiFi interface, the N900 integrates mobile network modules in it. It can support 2G (GSM/GPRS) and 3G (UMTS/HSDPA) network for data transmission. Therefore, we can choose and compare more wireless links in different link rates in our experiments by using N900. Furthermore, not only the improvement of hardware in N900, a more advanced OS, Maemo 5 [23], runs on it and brings more features. The browser of Maemo 5 on N900 is faster and can support more content types than the browser of N810. This is significant in our later experiment to get more and better data.

NI cRIO-9215 Analog Input Module

As an important component in our experiments, the NI cRIO-9215 [25] analog input module is the core device to measure the real-time power changes of our mobile device. The main purpose of using this device is to monitor and sample the fast

Figure 5: NI cRIO-9215 Analog Input Module

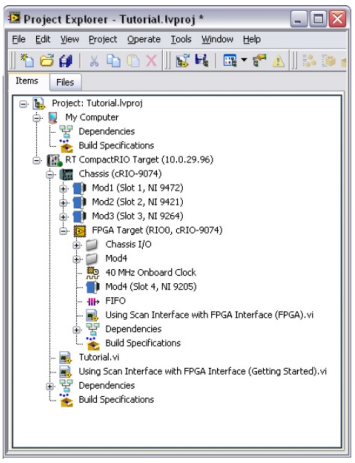

Figure 6: NI LabVIEW

fluctuant voltage on specific circuit component. It has four sampled analog inputs that can be used simultaneously. And its sample rate is 1000 samples per second, which means that it can sample one hundred thousand values of voltage per second. There is one supporting software, NI LabVIEW, for this device. User can connect this device through "RIO Scan Interface" directly to PC by using standard USB port, and monitor timely data graph in the software. Furthermore, by default the software displays voltage graphs. User can use advanced option to transform the output value into another one by user specific formula, for example, transforming voltage value into power value. After that, the software can then display graphs of new values and export these values into textual files. This software is convenient and will be used in later measurements.

3.2.2 Measurement Theory

The Figure 7 opens up the details of the measurement architecture in our experiments. There are four components in the system circuit: power supply, mobile device, one resistance, and one analog input module for voltage sampling use. The power supply, mobile device and resistance are series connected in the circuit. And the analog input module is parallel connected onto the resistance, in order to measure the voltage change on the resistance. However, our goal is to measure the timely

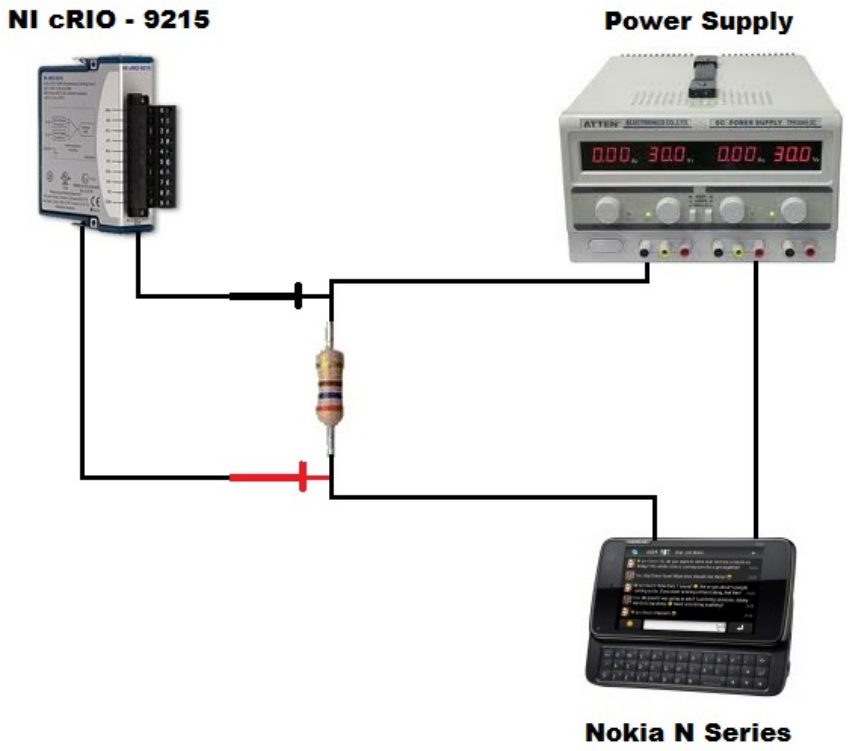

Figure 7: Measurement Devices

power consumption of the mobile device. Therefore, here we need to use electrical and mathematics knowledge to create a transform formula.

Let us first make sure which values are already known or can be measured during run-time in this infrastructure:

- Voltage provided by power supply: this value is constant, 3.9V in our experiment.
- Resistance value: this value is constant, 1.0 Ohm
- Instant voltage on resistance: this value changes with time, however, the analog input module can measure and record several instant voltage values based on its sample rate. Therefore, we can treat the instant voltage as constant value at every sample point.

Based on basic electrical knowledge, the power value (P) of device equals to product between current value (I) and voltage value (U) of device:

$$P = U \times I \tag{5}$$

Therefore, if we would like to obtain the instant power value of mobile device at every sample point, then we need to know both instant voltage value and instant current value of mobile device. As we mentioned before, due to the series connection between resistance and mobile device, the instant current value of mobile device is identical with the value of resistance. In addition, the voltage of mobile device equals to voltage of power supply minus voltage of resistance. As we know, current value (I) equals to the ratio between voltage value (U) and resistance value (R) of specific device:

$$I = \frac{U}{R} \tag{6}$$

Then we can rewrite the Formula 5 into:

$$P_D = (U_P - U_R) \times \frac{U_R}{R_R} \tag{7}$$

Here, P_D stands for the power value of mobile device; U_P is the voltage provided by power supply, which is 3.9; U_R is voltage value of 1.0 Ohm resistance; R_R is resistance value of resistance, which is 1.0 Ohm. Therefore, this formula can be finally simplified into:

$$P_D = (3.9 - U_R) \times U_R \tag{8}$$

The Formula 8 will be configured into the LabVIEW software. Then for each instant sample of the voltage on 1.0 Ohm resistance, we will get the instant power value of mobile device. And the software will log these values for later analysis use.

3.3 Summary

The purpose of the chapter 3 is to illustrating how we established the experiment environment for measuring the energy-efficient benefits brought by the new selective-

compression strategy. We introduced the application environment by explaining the structures and the operating principles of both client proxy and remote proxy into details. Then in order to show how we measured the results of each experiment, we presented the specifications of the mobile devices and measurement devices used in our experiments. At the end, we illustrated the formulas used for calculating the instant power values on mobile devices based on the basic electrical theory. Understanding these knowledge is necessary before we enter the experiment design chapter.

4 Experiment Designs

Until this chapter, we have already introduced measurement devices and measurement theory for our experiments. In this chapter, we will describe the design details of our experiments. However before that, we would like to clarify that the following experiments were done on two different mobile devices, the Nokia N810 and the Nokia N900. If noticing the introduction of N810 and N900 in previous chapter, we should know that the N900 was announced two years later after N810. We started our measurement before N900 announced, therefore, the N810 was the best choice for the first phase research about energy-efficient web content compression and transmission. Later we switched our device into N900 for second phase experiment after it came out, and run our client proxy on it for more complex experiment in real network. After understanding this, let us start to show the experiments.

Phase One

4.1 Experiment 1: Web Content Compression

From the Chapter 2.2.2, we know that there are various of web content types in web pages. And they can be grouped based on their characteristics. The result of earlier researches [3] [6] also revealed that the characteristics of file contents decide their compression ratios and whether they are worth to be compressed or not. Therefore in this experiment, we will measure the compression ratios of compressing real web pages, which are fetched from Internet, by using different compression tools.

By using the Alexa Traffic Rank [27] as the criterion, we chose and downloaded fourteen popular websites in Finland and worldwide for this experiment. Table 2 lists the size of all contents of each website's main page. The Alexa Traffic Rank can mainly reflect the amount of IPs reaching each website per day. The higher the Alexa Traffic Rank the website has, the more popular they are among the Internet users. All of these selected websites have high Alexa Traffic Rank in both global and their country. For example, "google.com" is rank No.1, and "facebook.com" is rank No.2 in both global and US. The Finnish website "iltalehti.fi" is rank No.1585

Table 2: Original sizes of web pages

Web name	Size (KB)	Web name	Size (KB)
Blogger	126.8	Iltalehti	2048
Google	84.6	BlueChillies	90.9
Twitter	165.4	TrendyFlash	453.6
Facebook	941.6	LikeMinds	1331.2
Adobe	541.9	CNN	980.5
Yahoo	812.9	Ebay	442.2
Microsoft	503.2	Apple	872.1

Table 3: Details of web page objects

Name	Text	Binary	Name	Text	Binary
Blogger	10	2	Iltalehti	19	136
Google	3	1	BlueChillies	2	11
Twitter	6	3	TrendyFlash	4	47
Facebook	25	2	LikeMinds	22	18
Adobe	26	11	CNN	44	101
Yahoo	12	12	Ebay	9	15
Microsoft	19	15	Apple	18	12

in global and rank No.5 in Finland. Therefore these websites are famous enough and can represent major types of modern web pages. Furthermore, the Table 3 lists the content details of each web page by grouping contents into two categories, text and binary. This table also reveals the general constitution of each web page based on the amount of text and binary files. For example, "Google" contains only 1 Graphics Interchange Format (GIF) file and 3 text files. Therefore "Google" mainly consists of textual data. Similar with "Google", the main page of "Facebook" is more textual based web page, which contains 25 text files but only 2 smail GIF files. The "LikeMinds" seems to be more moderate web page. It has almost same number of text and binaries. However, the "Iltalehti" is like a binary monster comparing to others. It consists of only 19 text files, but 136 multimedia binaries. These special web pages are excellent samples for proofing the different compression effects on different types of web pages.

The bzip2, gzip and lzma are the compression tools in this experiment. These compression tools have good performance for compressing most file types. Even the compressed binary files, such as GIF can be further compressed by these compression tools. [6] Therefore, we manually separated web contents into two groups, text and binary. And we proceeded the experiment by using two compression strategies:

– compressing both text files and binary files

– compressing text files only and leave binary files uncompressed

In this experiment, we named the first strategy as "all-compression", and the second one as "textual-compression".

4.2 Experiment 2: Compression Effect to Varied Bandwidth Data Transmission

Our Experiment 1 generally reveals the effect while applying different compression tools on various web page contents which have different total size and different file type constitution. Therefore in this experiment, we illustrates the real effect of compressing web contents and transmitting them over wireless links with varied bandwidth when receiving web contents on mobile device.

The Figure 8 presents the experimentation environment. There are three important components in this environment: Nokia N810, proxy server, and the D-Link WLAN wireless router.

The Nokia N810 acts as the mobile device. And our test client proxy runs on it. We use its local web browser as the test mobile browser. And it is configured to communicate with the test client proxy. When the web browser requests web contents from client proxy, client proxy forwards requests to remote proxy, and waits for response. The response data will be decompressed if necessary and be passed back to web browser by client proxy.

The remote proxy server has responsibility to forward requests from mobile device to Internet web servers, and to return compressed or uncompressed content data back to the mobile client.

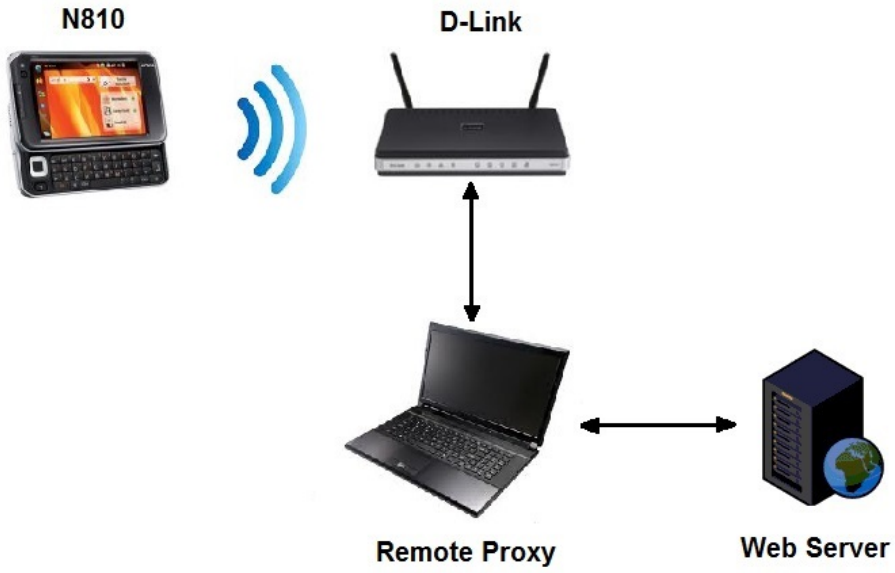

Figure 8: Architecture of Experiment 2

Because that the Nokia N810 does not have mobile wireless interfaces (2G, 3G), in this experiment we simulated varied bandwidth environment by controlling the data rate of WiFi connection. We imported D-Link wireless router into experiment as wireless access point. And, we managed to adjust the transmission bit-rate of both D-Link wireless and wireless interface on proxy server, which runs Ubuntu OS, from 1Mbits/s to 54Mbits/s. These different link-rates stand for varied bandwidth here. One issue is that the WLAN interface on N810 has power-saving feature. Generally this feature is to temporarily shutdown the WLAN interface for energy-saving. However, in our measurement, we only want to test the energy-saving effect brought by the new compression strategy. The power-saving feature might interfere the result of our measurement, because that when no data transfer is ongoing, this feature might bring additional energy-saving to the N810. Therefore, we disabled the power-saving feature of the WLAN interface on N810.

Typically, the maximum throughput of mobile wireless interface is less that its theoretical maximum value. Therefore, We proceeded a preliminary test before starting this experiment, in order to measure the practical downlink capability of WLAN interface of the N810. We used software namely Jtg [26] as packet sender to

send udp or tcp packets with specific packet length and interval on remote computer, and as receiver to receive these packets on N810. This software can display statistics information of link bandwidth when finishing data transmission. And from the result of receiving udp packets at high speed on N810, we figured out that the real maximum receiving capability of N810 is around 4.5Mbits/s, which is far away from its theoretical throughput, which is 54 Mbits/s. Therefore, we understood that in this experiment we need to adjust transmission link bandwidth to proper values that can reveal the compression effect. It does not make any sense to send data to N810 at high speed, for example 54Mbits/s, because that the maximum receiving rate is only 4.5Mbits/s, no matter how fast you sent to it. Therefore, in order to measure reasonable results from experiment, we adjusted the bandwidth between D-link wireless router and proxy server into four representative transmission data rates: 1Mbits/s, 2Mbits/s, 5.5Mbits/s and 18Mbits/s.

From the web pages listed in Table 2, we selected three representative web pages for this experimentation, "Facebook", "LikeMinds", and TrendyFlash". We compressed these page contents on the remote proxy server, based on the "textual-compression" strategy mentioned in Experiment 1, and decompressed contents by client proxy before returning requested web contents back to web browser. All information of the energy consumption and transmission duration will be monitored on mobile device side for later analysis.

Three different compression methods were selected for this experiment, HTTP compression, zlib, and lzo. Here we configured the HTTP compression to adapt gzip with default compression level (level 6) as its compressor. As mentioned in Chapter 2.1.3, HTTP compression uses textual-compression strategy by default. Therefore, other compressors, zlib and lzo, are all configured to use their lowest compression level (level 1) and ignore binaries as well. In this way, we can clearly reveal the effect of energy-efficiency by using compression methods on different web pages through various link bandwidths.

Phase Two

4.3 Experiment 3: Selective Compression Assisting Energy-Efficient Web Access

Before we designed this experiment, the Nokia N900 smart phone was announced. Due to it carrying with mobile radio modules (2G, 3G), we decided to conduct experimental tests by applying the energy-efficient proxy-pair onto real mobile Internet, in order to analyze the real effect of selective data compression for web browsing on mobile device. We supposed to compare the performances of energy consumption and delay for accessing specific web pages with our proxies with the performances without our proxies.

The Figure 9 describes the architecture of the Experiment 3. Its basic structure is similar with the Experiment 2, with few main differences:

- instead of only using WLAN to simulate mobile radio transmission, we used real 2G, 3G mobile data connections and WLAN in this experiment.
- Ideally, the remote proxy should be deployed and run by ISPs or network operators on there mobile core network, and as close as possible to mobile client so as to minimize transmission delay between mobile device and proxy. However we can not simply add proxy feature to any operator. Therefore, in order to simulate similar environment, we installed our remote proxy to a server machine, which was connected on the bone network of our campus. This server machine is Unix based and powerful enough to compress files in a short moment, and transmit them to mobile devices with minimum delay.

The Figure 10 is the operation flow chart of our experiment environment. As shown in it, the mobile browser initiates a HTTP GET request and forwards request to client proxy. The client proxy acts as transparent proxy at this phase, and forwards original request to the remote proxy. The remote proxy transparently forwards the request to the web server. After remote proxy fetches target content from web server, it selectively compresses this data based on specific compression

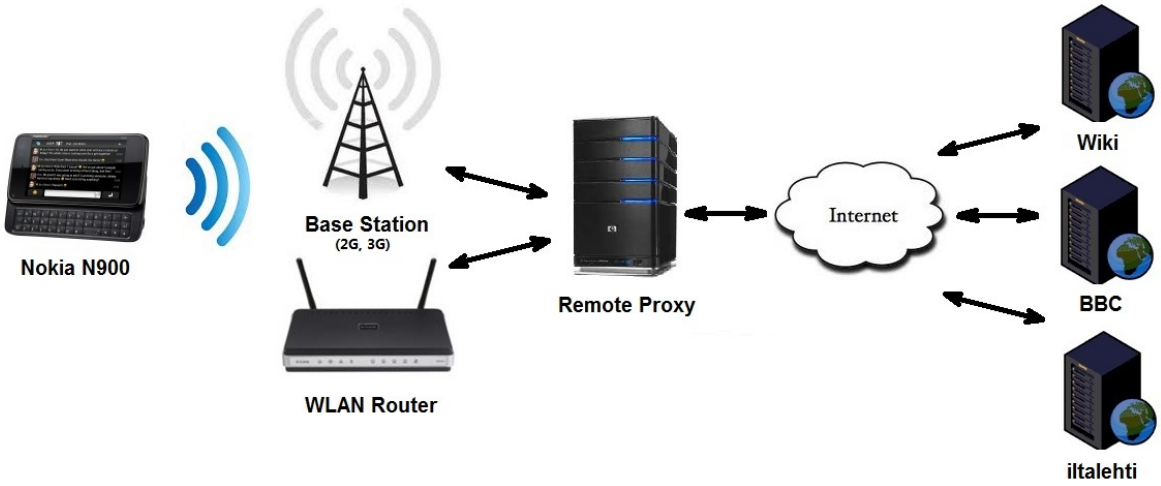

Figure 9: Structure of Selective-Compression Experiment

ratio mentioned in Table 1. Then after finishing selective-compressing, remote proxy responses the selected and compressed data back to client proxy. The client proxy manages to decompress received data if necessary, and finally, the original content data will be passed back to web browser, and shown to mobile user.

Similar with Experiment 2, we selected three representative web pages, "Wikipedia", "BBC", and "Iltalehti" for this experiment. Furthermore, we controlled the RTT of fetching web pages by using Linux netem [24] to add extra delay between remote proxy and web servers, in order to simulate different distance from web server in the world and then evaluate the impact of link quality or distance to energy consumption. For example, the web server with 30ms delay means domestic web server; with 150ms delay means international web server; and with 400ms delay means intercontinental web server. Thus, we can make our experiment environment more close to realistic situation.

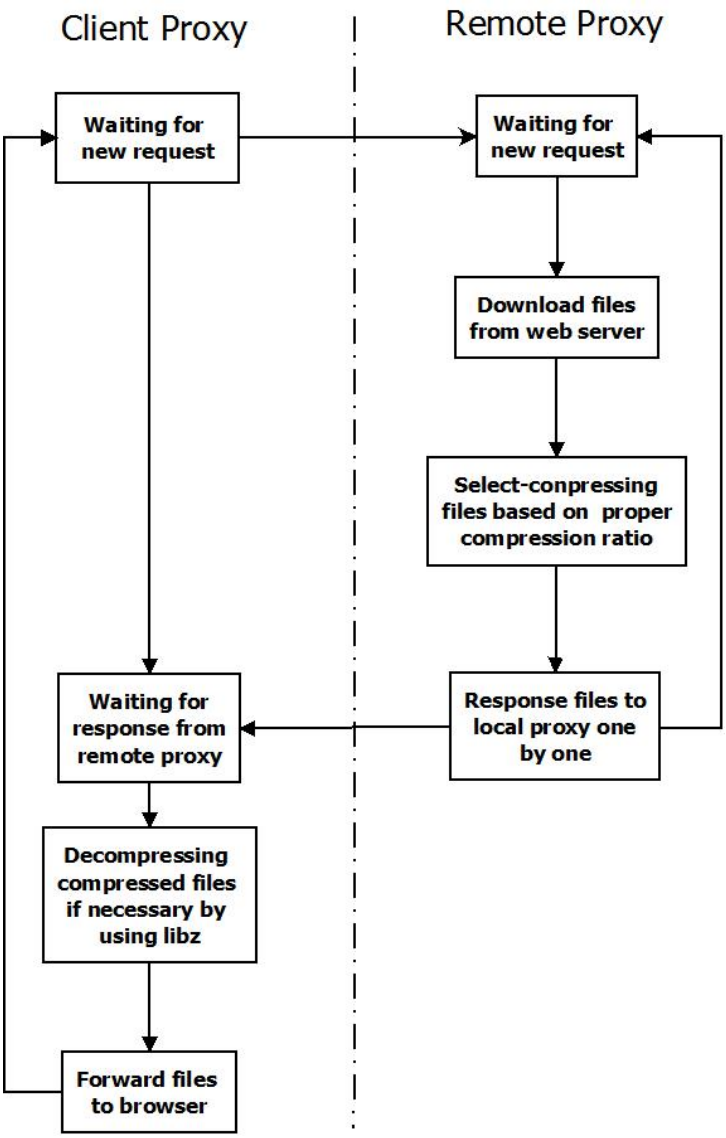

Figure 10: Flow Chart of Selective Compression Experiment

5 Results and Analysis

In this chapter, we present the results got from our experiments, and separately analyze these results in order to reveal facts behind them.

5.1 Experiment 1

The Figure 11, 12, and 13 present the results of this experiment in general. And the Table 4 shows the numeric details of the results. As shown in these figures and table, all web pages can more or less obtain benefit from compression. In order to show results clearly, we divided the final results of compressing these web pages into three groups by different phenomenons after using different compression strategies. The Figure 11 shows the group of web pages which got almost similar compression ratios (only about 1% size difference) by using all-compression and textual-compression strategies. Figure 12 shows the web pages that obtain more differences among using different compression strategies. For example, the size of "Iltalehti" after applying all-compression strategy by using bzip is more than the size after applying textual-compression, but less by using gzip and lzma. Figure 13 shows the group of web pages that gain more compression ratios by using any compression tool for both all-compression and textual-compression. If we compare the results in Figure 13 carefully, we can notice that there is one interesting web page namely "BlueChillies". The total size of "BlueChillies" after using all-compression strategy is 50% less than the total size after using textual-compression strategy. The similar phenomenon happens on the web page "LikeMinds" when using lzma as compression tool. Its size after using all-compression strategy is clearly less than the size after using textual-compression strategy.

If we observe the results by comparing to the amount and type of objects in each web page, we find a clear relationship between them. As known to us that most of binary files on Internet have been in compression format already, we can well compress web pages that contain very few binaries without specially handling these binaries. However, when the amount of binary files is large and even exceeds the

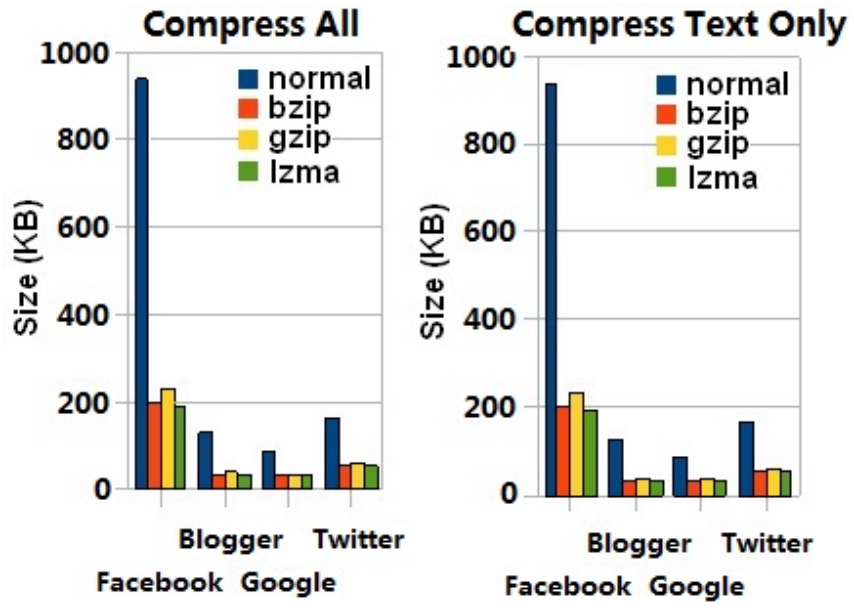

Figure 11: Web Pages with Slight Size Difference Between All-compression and Text-compression

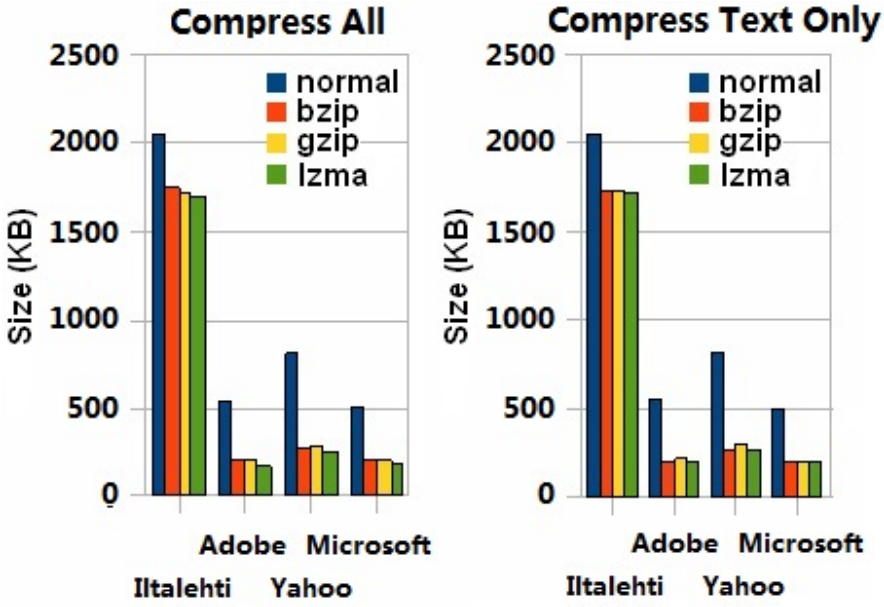

Figure 12: Web Pages with Various Changes by Using Different Compressors between All-compression and Text-compression

amount of text files (like BlueChillies), then we must take binary compression into consider. This is because that textual-compression strategy is insufficient in this situation. And compressing binaries can bring more compression ratio than only compressing text files.

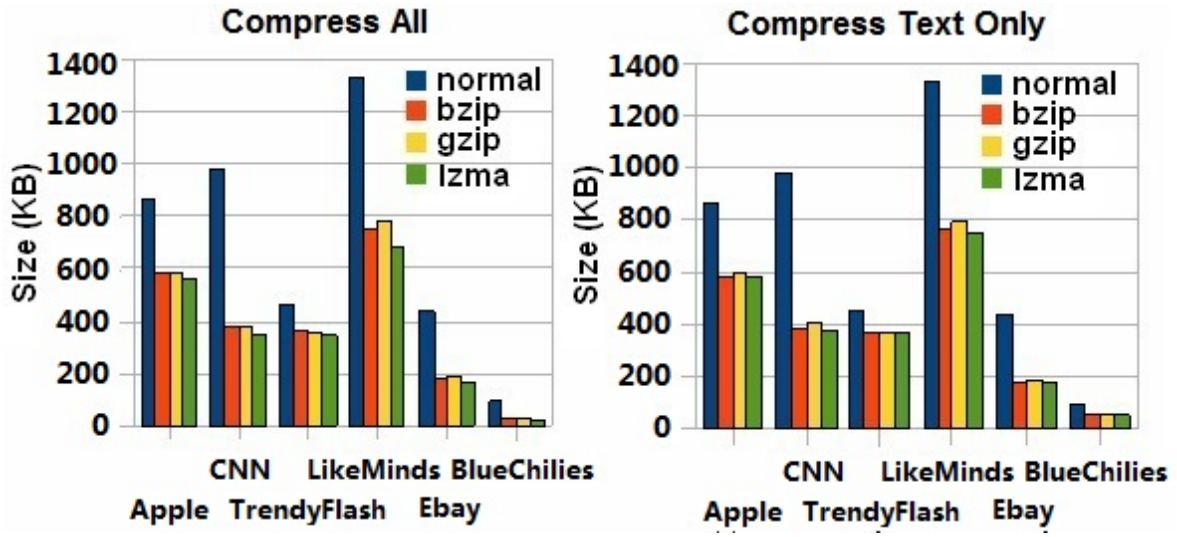

Figure 13: Web Pages with Size Decreasing by All-compression Compared to Text-compression

Concept of "Green" Web Page

In order to summarize the results of this experiment, we would like to define a new concept, in order to group web pages based on the energy-efficiency web access. We call it the concept of "Green" web page. In general, we define the "Green" web page as the one that can be delivered with minimal amount of bytes. Such web page must have characteristics of being well designed without redundant information, and being easily accessed by mobile devices with limited download link speed. From the perspective of compression, a "Green" web page can also be defined by how well the web page contents can be compressed. More bits we can save by compression, more "Green" the web page is. Corresponding to "Green", the "Black" web pages are those not well-designed, and contain massive redundant or hard-to-compressed information.

We rearranged the experiment result shown in Table 4 and assigned them into three groups in the Table 5, in order to reveal how "Green" these various web pages are. The first group shown in the top subtable contains "Green" web pages. As the representation, the "Facebook" achieves the highest compression ratio, 79.65%, by using lzma. Even the lowest compression ratio that belongs to "Blogger" by using gzip in this group still reaches 71.61%. The bottom group is "Black" group. All web

Table 4: Compression sizes of web pages by using all-compression and textual-compression strategies (Size in KB)

Name	Normal	Strategy	bzip	gzip	lzma
Facebook	941.6	compress all	197.5	233.6	191.6
		compress text only	197.3	233.4	191.8
Blogger	126.8	compress all	35.1	36.6	32.6
		compress text only	34	36	32.5
Google	84.6	compress all	35.9	36.2	33.4
		compress text only	33.9	35.6	33.3
Twitter	165.4	compress all	56	58.1	52.3
		compress text only	53.8	56.4	52.5
Iltalehti	2048	compress all	1753.8	1725.7	1700
		compress text only	1725.1	1734.3	1718.9
Adobe	541.9	compress all	200.4	208.7	183
		compress text only	199.8	212.5	193.4
Yahoo	812.9	compress all	271.2	289.9	256.6
		compress text only	266.4	290.9	262.7
Microsoft	503.2	compress all	201.3	203.8	188.5
		compress text only	195.4	204.5	192.1
Apple	872.1	compress all	584.1	582.9	562
		compress text only	586.1	594.4	581.5
CNN	980.5	compress all	370.9	377.5	342.5
		compress text only	389.7	409	381
TrendyFlash	453.6	compress all	359	352.8	345
		compress text only	362.4	363.9	361.6
LikeMinds	1331.2	compress all	748.2	780.3	683.4
		compress text only	761.5	791.5	751.9
Ebay	442.2	compress all	179.4	185.8	167.5
		compress text only	182.5	190.7	174.9
BlueChillies	90.9	compress all	24.3	25.3	20.9
		compress text only	49.5	50.1	49.4

pages in this group can hardly be compressed by any compressor, such as "Iltalehti". The best compression ratio of this web page is only 15.74% by using gzip.

If we look back at Table 3, we can figure out some potential relationships between the file type consistent of web contents and how "Green" web page is. For example, the "Facebook" and "CNN" are proven to be "Green" web pages, because that they contain more text files then binaries. However, "Iltalehti", which has huge amount

Table 5: Best compression ratios of web pages (Size in KB)

Name	Normal	bzip	gzip	lzma
Blogger	126.8	34 (73.19%)	36 (71.61%)	32.5 (74.37%)
Facebook	941.6	197.3 (79.05%)	233.4 (75.21%)	191.6 (79.65%)
BlueChillies	90.9	24.3 (73.27%)	25.3 (72.17%)	20.9 (77.01%)
Adobe	541.9	199.8 (63.13%)	208.7 (61.49%)	183 (66.23%)
Yahho	812.9	266.4 (67.23%)	289.9 (64.34%)	256.6 (68.43%)
CNN	980.5	370.9 (62.17%)	377.5 (61.5%)	342.5 (65.07%)
Twitter	165.4	53.8 (67.47%)	56.4 (65.9%)	52.3 (68.38%)
Google	84.6	33.9 (59.93%)	35.6 (57.92%)	33.3 (60.64%)
Ebay	442.2	179.4 (59.43%)	185.8 (57.98%)	167.5 (62.12%)
Microsoft	503.2	195.4 (61.17%)	203.8 (59.5%)	188.5 (62.54%)
LikeMinds	1331.2	748.2 (43.8%)	780.3 (41.38%)	683.4 (48.66%)
Iltalehti	2048	1725.1 (15.77%)	1725.7 (15.74%)	1700 (16.99%)
Trendy Flash	453.6	359 (20.86%)	352.8 (22.22%)	345 (23.94%)
Apple	872.1	584.1 (33.02%)	582.9 (33.16%)	562 (35.56%)

of binaries in its content files, must be "Black" web page with no doubt. Therefore, improving the access efficiency of "Green" web pages by compressing is easier than "Black" ones. And downloading "Black" web pages brings great burden for mobile devices in energy consumption, and long access delay aspects. User has to download redundant binaries before they can find their interested information.

Furthermore, the definition is not that simple. We could claim that, for example, the "Ebay" is "Greener" web page than "Facebook" from the perspective of energy-efficiency. Although the compression ratio of "Ebay" is less than "Facebook", however it decreases more considerable amount of data size by using compression than "Facebook". Therefore, essentially, the ultimate criterion to define energy-efficient "Green" web pages is, how much data transfer a certain web page requires, no matter with or without compression. For example, the "Apple" without compression is still much more "Green" than "Iltalehti" with highest compression ratio, because that it still has less size and consumes less energy for data transfer overall in the network than compressed "Iltalehti".

5.2 Experiment 2

In the discussion of Chapter 2.2.1, we mentioned that there are several operations consuming battery energy on mobile devices while receiving data via wireless interfaces. For receiving uncompressed data, the energy consumption consists of consumption of computation and wireless interface. And for receiving compressed data, the energy consumption is the sum of operations for receiving, storing, and decompressing data. The Figure 14, 15 and 16 show us the results of the Experiment 2 from two different aspects, energy consumption and delay. They reveal the benefits brought by compressing specific web pages with different compressors under different bandwidths of wireless links.

Energy Consumption

In order to quantify the energy efficient brought by using data compression, we use the "energy-saving ratio" to measure it. The "energy-saving ratio" equals to the "original energy consumption of file transfer" without using data compression divided by the "decreased energy consumption of file transfer" by using data compression.

The "Facebook" is considered as "Green" web page in Table 5. This conclusion is confirmed by the experiment results shown in Figure 14 as well. This figure shows the energy-saving performance of "Facebook" by using text-compression strategy. And it clearly shows that this strategy saves considerable amount of energy when accessing "Facebook". When using HTTP compression with slow link, 1Mbits/s, the energy-saving ratio is 62.7%, which is very high. This ratio becomes 42.1% even when the link bandwidth raises to 18Mbits/s, and still good enough. The Figure 15 shows the experiment results of "LikeMinds". Not like "Facebook", "LikeMinds" is a web page that in the middle of "Green" and "Black", which means that besides text files, it also contains almost same amount of binaries. This web page gains 29.4% energy-saving when bandwidth is 1Mbits/s and the HTTP compression is in use. However, after the link bandwidth goes upto 18Mbits/s, this ratio decreases to only

20%, but still acceptable. However, the results of "Black" web page "TrendyFlash" shown in Figure 16 are not as good as previous ones. The best energy-saving ratio that it can reach is only 23% by using HTTP compression at 1Mbits/s low bandwidth link. And the other two compressors almost can not even achieve any energy-saving benefit at this bandwidth. Only about 5% energy can be saved in "TrendyFlash" by zlib and lzo compression with lowest compression level when using text-compression strategy.

From the phenomenons shown in Figure 14, 15 and 16, it is explicit that the HTTP compression performs better either for "Green" or "Black" web pages for energy-saving than zlib and lzo with lowest compression level. Previous researches [3] [6] have already figured out the trade-off between the compression level and energy consumption: the more data we would like to compress, the more energy will be consumed. However, in our results, the compressor with higher compression level (more processing) performs much better than compressors with lower compression level (less processing). This phenomenon indicates that on mobile device, the energy saving from less data transmission outweighs the energy consumption increased by CPU processing and memory accessing. Therefore, here we can confirm that it is always worth to spend extra energy on CPU cycles and memory for reaching better compression ratios, in order to save energy.

Delay

The Figure 14, 15 and 16 also present the changes of delay while receiving certain web contents under different bandwidths. It is obvious that the "Green" web pages can still obtain more benefits in decreasing transmission delay than "Black" ones by using text-compression strategy. In Figure 14, the "Facebook" can be downloaded 10 seconds faster when using HTTP compression under 1Mbits/s bandwidth, and 2 seconds less even under 18Mbits/s. Downloading time of "LikeMinds" shown in Figure 15 is also decreased by 6 seconds by using HTTP compression with 1Mbits/s bandwidth, and by about 2 seconds with 18Mbits/s. As supposed, the "Black" web page "TrendyFlash" in Figure 16 can only save 3 seconds by using HTTP

compression with 1Mbits/s link, and about 1 second less with 18Mbits/s bandwidth. Results become even worse when using zlib and lzo. "TrendyFlash" can not gain any time deduction from compression any more.

There is also a trade-off between time consumed by decompressing data and the time saved by less data transmission. Our analysis indicates that the time decreased by compressing more data exceeds the time increased by data decompression. Therefore, we confirm that higher compression level can help more to reduce transmission delay.

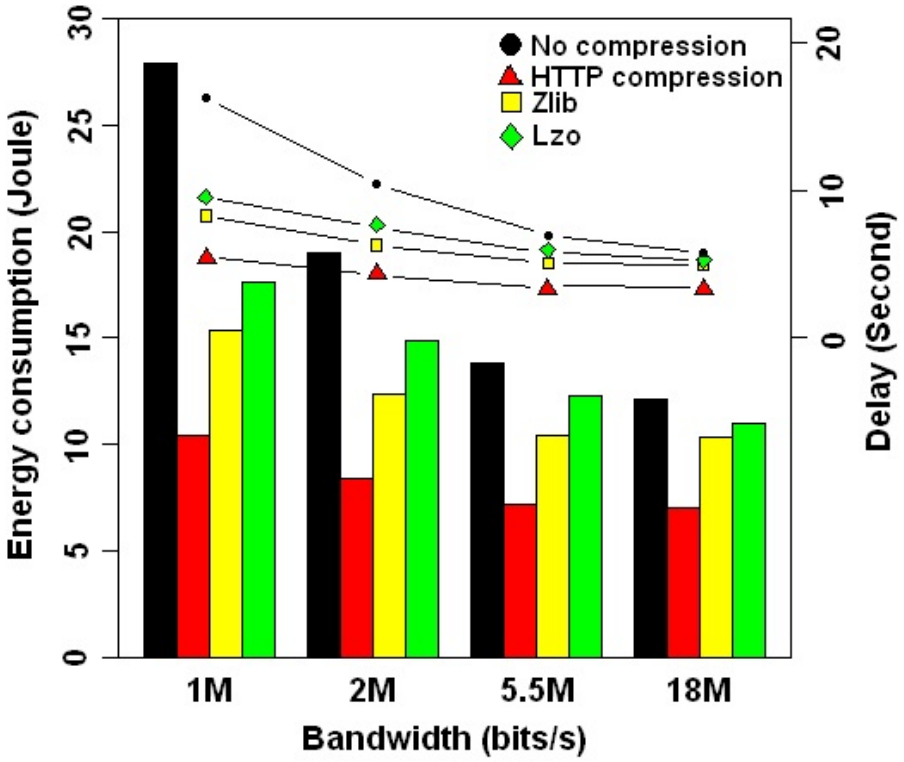

Figure 14: Energy Consumption and Delay of "Facebook"

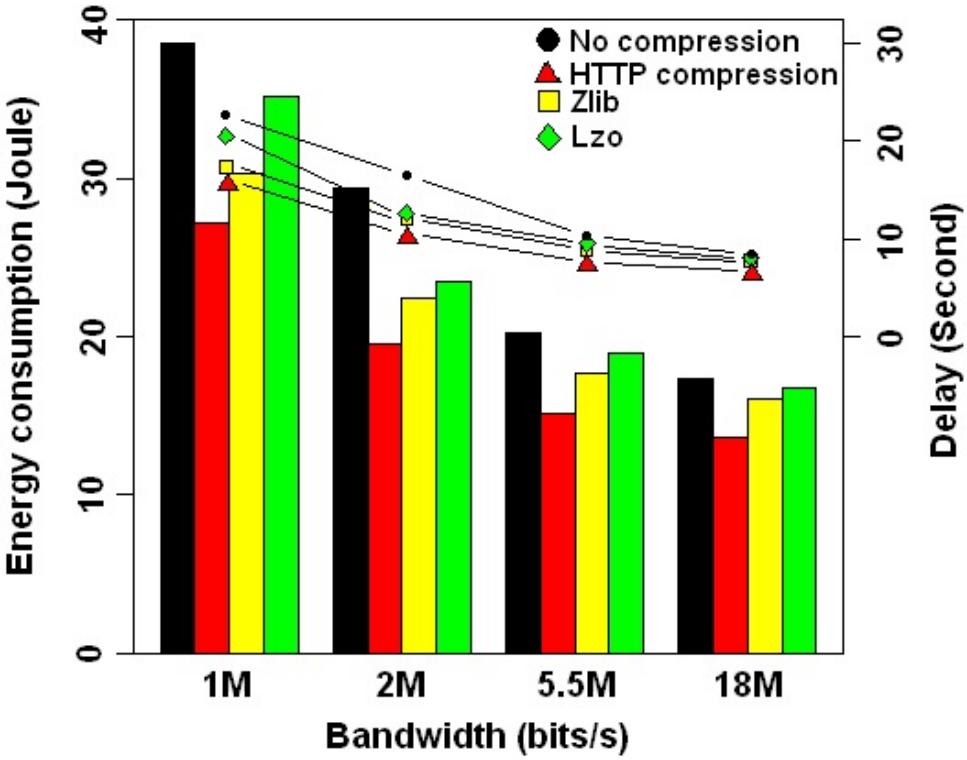

Figure 15: Energy Consumption and Delay of "LikeMinds"

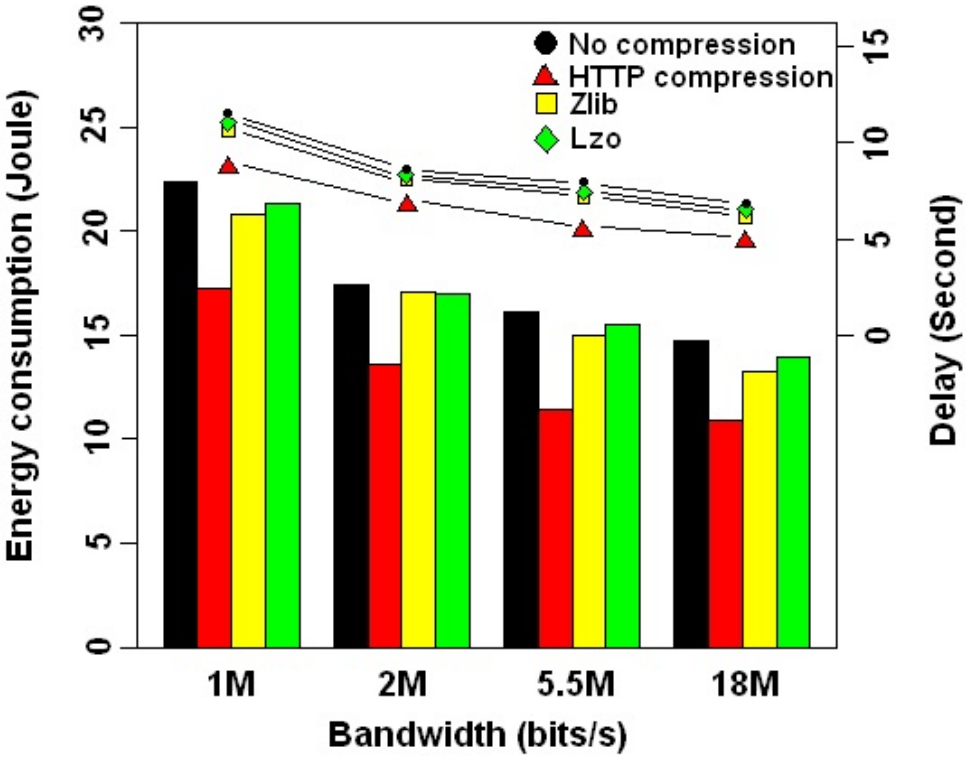

Figure 16: Energy Consumption and Delay of "TrendyFlash"

5.3 Experiment 3

The results of Experiment 2 reveal one issue, that is, the traditional text-compression strategy is insufficient when dealing with "Black" web pages to both save energy and deduct delay. Therefore, in this experiment, we applied our new compression strategy for web accessing. Since we defined the concept of "Green" web page based on the results of Experiment 1, then we could tag the three web pages used in Experiment 3 as "Green" (wikipedia), "Normal" (BBC), and "Black" (iltalehti) web pages.

The Figure 17, 18, and 19 present the performances of downloading these pages with or without selective-compression via different mobile networks. The Figure 17 shows that generally selective-compression offers great power saving and short delay when downloading all three web pages via 2G network. The average energy saving and delay deduction are both around 15% by adopting the compression strategy. This result is reasonable as it should be, due to the slow link speed and high energy consumption per bit of 2G network. This result also proves our assumption made in Chapter 2.1.2: the lower link speed and higher energy consumption per bit, then the more benefits the data compression can offer. However, after switching to 3G network that has higher link speed and less energy consumption per bit, the energy saving benefits brought by our new strategy become unclear.

The results shown in Figure 18 reveal that the selective-compression does not bring any benefit to save energy and deduct delay for downloading "Green" and "Normal" web pages. Notice that while fetching "Wikipedia" with 400ms delay as an example, the mobile device even consumes about 5% more energy with selective-compression. This is because that with higher link speed, the effect of data size deduction by compression for "Green" web pages becomes minor factor. And delay becomes major factor for energy consumption because it affects the active time of mobile wireless interfaces. Therefore, the extra delay brought by selective-compression logic will cause more energy consumption on "Green" web pages. However, our new strategy keeps performing well for "Black" web page "Iltalehti", especially when downloading "Iltalehti" with 400ms delay, the energy-saving ratio reaches to about

15%. And the delay is reduced by almost 20%.

The results when using WLAN network in this experiment shown in Figure 19 are similar with the results of 3G. Our new compression strategy becomes more of a hindrance than a help to "Green" and "Normal" web pages. For "BBC", selective-compression causes 10% more energy consumption than normal web accessing without compression. However, even with very high link speed, our strategy can still offers considerable benefits for energy-efficiently downloading the web contents of "Iltalehti". Despite that there isn't any delay deduction, anyhow the "Iltalehti" still obtains about 5% energy saving on average by using selective-compression. All these successes of handling "Black" web page with different link speeds further prove our strategy is great useful for energy-efficiently web accessing to "Black" web pages.

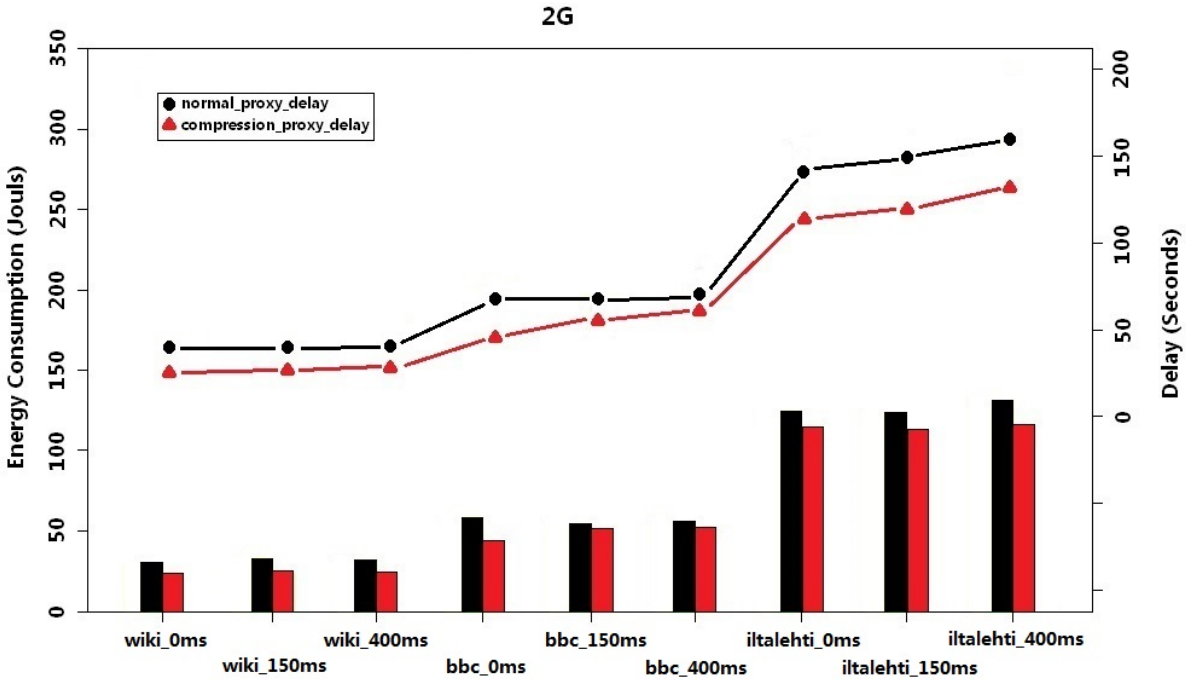

Figure 17: Power Consumption and Delay of Selective-Compression in 2G network

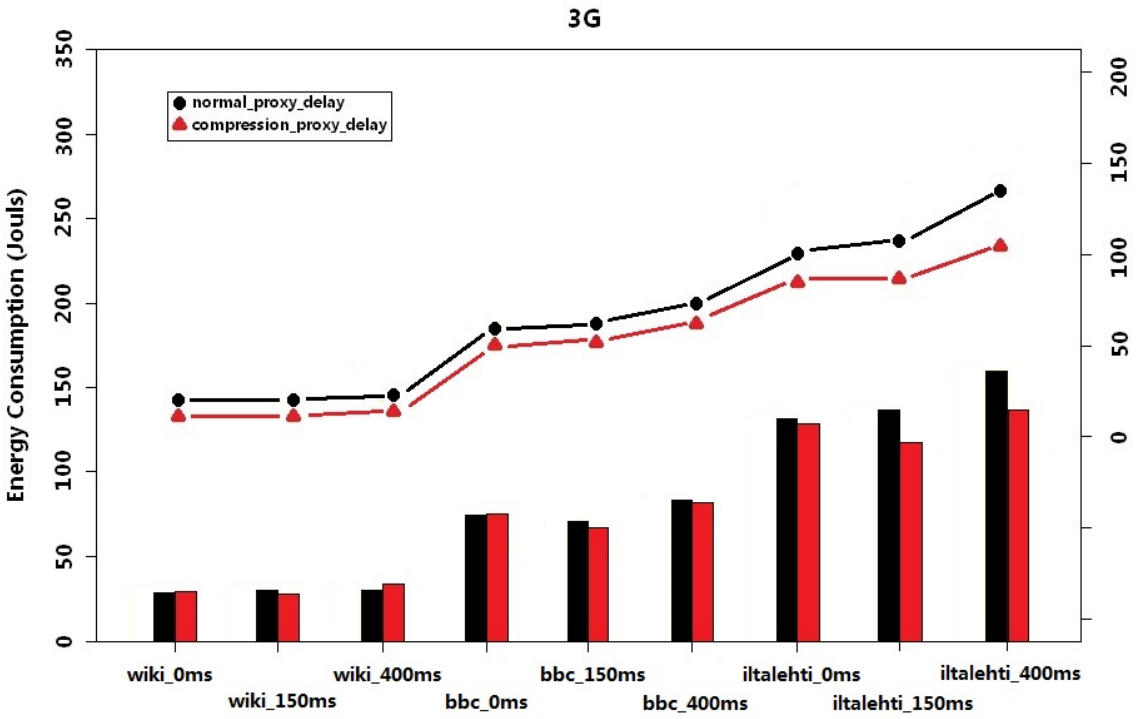

Figure 18: Power Consumption and Delay of Selective-Compression in 3G network

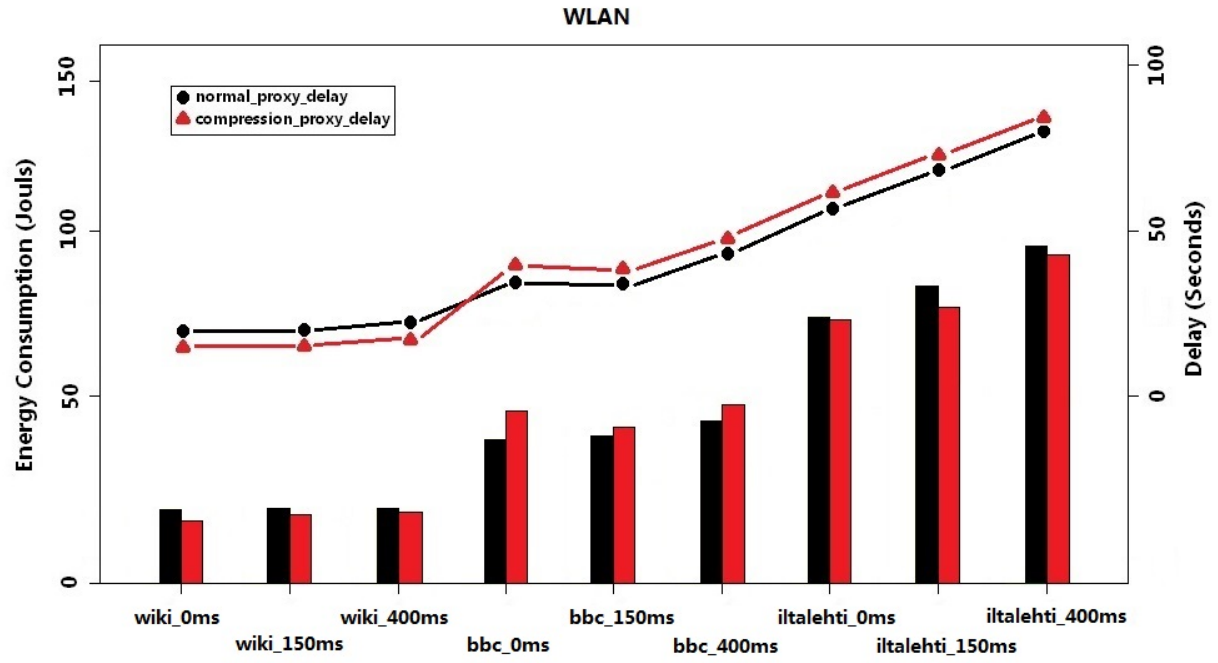

Figure 19: Power Consumption and Delay of Selective-Compression in WLAN network

5.4 Discussion

It is obvious that the results of our experiments are remarkable and could be used for improving energy efficient web access. Especially under relatively low bandwidth wireless connection, implementing our new selective-compression strategy can save considerable energy and decrease data transfer delay when accessing web pages on mobile devices. The operators can use our new solution to provide new energy efficient web access service to their end users by establishing selective-compression proxies on their bone network. However operators should notice that our new strategy brings energy saving to end users by consuming more energy on remote proxy. The less the data transferred from operator's proxies to end users by using selective-compression strategy, the more energy consumption on them. When dealing with huge amount of web requests and selective-compression tasks, the energy consumption of proxies must be considered.

Besides the side-effect of energy consumption on proxies, the selective-compression strategy has another limitation. As we mentioned in Chapter 2.2.1, the energy-efficient compression ratios are used as the criteria of selective-compression strategy to select compressible files. However, these ratios are different among operators because of their different network configurations. Therefore, it is impossible to hard code these ratios in our proxies by default. These ratios must be measured and calculated by different operators offline, and manually be applied to each proxy as predefined parameters before proxy being launched.

Furthermore, after many analytic works about the consistent and characteristics of web contents, the current web page development becomes a little worry for us. Because of more and more requirements of fetching rich contents from web suppliers, nowadays there are more and more monster level web pages, like "Iltalehti", daily accessed by mobile users. These web pages contain huge amount of binaries with big sizes (megabytes level usually). By using wireless data links, especially low speed data links, mobile devices spend massive of battery energy for downloading web contents before the useful information appearing to user. Although we can apply the new solution on several web servers or proxies of operators so that they are able

to offer energy-efficient web browsing experience to many mobile users, this is still far away from enough. Therefore here we address the cause, not only the symptoms. As an important conclusion from the view of "Green" web page concept, we would like to provide suggestions for designing and developing future web pages for mobile devices in energy-efficient manner:

- deducting the amount of binaries in each web page
- energy-efficiently compressing binaries if necessary
- maintaining mobile version of each web page which only contains optimized web structure and energy-efficient data contents

6 Conclusion

In this final chapter, we make the conclusions of this research work. First we present the summary of our research work based on the experiment results. Then we give an introduction of the future works of our research.

6.1 Summary

As the title of this thesis, the main goal of this thesis is to analyze and prove the feasibility of various approaches for achieving energy-efficient web access. To reach this goal, we designed three experiments in Chapter 4, and measured them with representative web pages. Finally, by analyzing all the results from these experiments, we arrived at following conclusions:

- For "Green" web pages, textual-compression strategy is always sufficient, due to less binaries contained in such web pages. However, for "Black" web pages which consist of massive well-compressed or uncompressed binary files, there is still room for data compression.

- The traditional textual-compression strategy is proper to save energy and reduce transmission delay for "Green" and "Normal" web pages. But it can only contribute little benefit to "Black" web pages. Along with the increasing of link bandwidth, the benefits obtained by compression decrease. Furthermore, the compression level of compressors is an essential factor for energy-efficient data compression. Selecting proper compression level, which has the better trade-off balance between transmission energy saved by compression and local energy consumption by decompression, can bring more benefits of energy-efficiency and delay reduction, for example, the gzip with level 6 used in HTTP compression by default.

- With low speed links, like 2G network, the selective-compression strategy performs great and contributes much to the energy-saving and delay-reducing

while accessing either "Green" or "Black" web pages. Especially for accessing "Black" web pages, the new strategy proves the advantages of selective-compressing binaries in order to achieve more energy saving and less delay, no matter with fast or slow wireless links. However, it is inevitable that the new strategy on remote proxy brings redundant time spend because of the selection and compression operation. The fact is that with fast wireless links (3G, WLAN), applying textual-compression on "Green" or "Normal" web pages performs better than selective-compression. Therefore, the hybrid compression strategy can be applied to obtain more energy-efficiency and shorter transmission time.

6.2 Future Works

This thesis work is just the beginning for reaching the goal of energy-efficient web access on mobile devices. Our testing software is still in prototype phase. Although we got exciting results from our experiments, our prototype solution is still far away from product level. We need to put more effort in our following research work to improve the algorithm performance of the new selective-compression strategy, and also improve the data transmission performance between the mobile devices and the remote proxy. Generally, we already have some main targets as our future work:

- On the remote proxy, we implemented the selective-compression algorithm in the way of trying to compress all web contents and selecting out compressible files based on the energy efficient compression ratios. We didn't care about how web pages are structured. In our future work, besides the compression ratios of web contents, we can import new criteria of selective-compression strategy for selecting compressible files based on the details of web page structures, in order to improve the selection performance of our compression strategy.

- Currently, the data transmission between the mobile devices and the remote proxy is inefficient. Each HTTP request from the mobile device can be only used to fetch one target data file in the HTTP response. In our future work,

we can apply better data transmission pattern between mobile devices and the remote proxy to improve data transmission performance. One efficient approach is to use "data bundling" on the remote proxy side. One HTTP request for accessing certain web page from mobile device can trigger the remote proxy to fetch the entire web page contents and bundle them together in one file. The mobile device will only receive one HTTP response including a bundling file which contains the whole target web contents.

References

[1] Gauthier, P., Harada, D. and Stemm, M. *Reducing Power Consumption for the Next Generation of PDAs It's in the Network Interface.* Power (mW), 1996

[2] Balasubramanian, N., Balasubramanian, A. and Venkataramani, A. *Energy Consumption in Mobile Phones: A Measurement Study and Implications for Network Applications.* Proceedings of the 9th ACM SIGCOMM conference on Internet measurement conference, ACM, 2009, pp.280-293

[3] Barr, K. C. and Asanovic, K. *Energy-aware lossless data compression.* ACM Transactions on Computer Systems (TOCS), vol. 24, no. 3, 2006, pp. 250-291.

[4] Xu, R., Li, Z., Wang, C. and Ni, P. *Impact of Data Compression on Energy Consumption of Wireless-Networked Handheld Devices.* Distributed Computing Systems, 2003. Proceedings. 23rd International Conference on. IEEE, 2003. pp.302-311.

[5] Maddah, R., Sharafeddine, S. *Energy-Aware Adaptive Compression Scheme for Mobile-to-Mobile Communications.* Spread Spectrum Techniques and Applications (ISSSTA '08). IEEE 10th International Symposium on, August 2008, pp.688-691.

[6] Wang, L. and Manner, J. *Evaluation of Data Compression for Energy-aware Communication in Mobile Networks.* Cyber-Enabled Distributed Computing and Knowledge Discovery, 2009. CyberC'09. International Conference on. IEEE, 2009.

[7] Wang, L. Yu, B. and Manner, J. *Proxies for Energy-Efficient Web Access Revisited.* Proceedings of the 2nd International Conference on Energy-Efficient Computing and Networking, ACM, pp55-58, 2011

[8] Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., and Berners-lee, T. *Hypertext Transfer Protocol - HTTP/1.1. RFC 2616. Internet Engineering Task Force(IETF).* 1999.

[9] Sillasto, E. and Wang, L. and Manner, J. *Using compression energy efficiently in mobile environment.* Green Computing and Communications (GreenCom), 2010 IEEE/ACM Int'l Conference on & Int'l Conference on Cyber, Physical and Social Computing (CPSCom). IEEE, pp9-16, 2010.

[10] gzip . [Online] [Cited: 13 01 2013.]
`http://www.gzip.org/`

[11] gwt. [Online] [Cited: 04 01 2013.]
`http://www.google.com/gwt/n`

[12] Opera Mini. [Online] [Cited: 24 01 2013.]
`http://www.opera.com/mobile/`

[13] OBML. [Online] [Cited: 24 01 2013.]
`http://dev.opera.com/articles/view/opera-binary-markup-language/`

[14] Lossless Compression. [Online] [Cited: 24 01 2013.]
`http://www.data-compression.com/lossless.shtml`

[15] zlib. [Online] [Cited: 16 01 2013.]
`http://www.zlib.net/`

[16] lzo. [Online] [Cited: 24 01 2013.]
`http://www.oberhumer.com/opensource/lzop`

[17] lzma. [Online] [Cited: 24 01 2013.]
`http://www.7-zip.org`

[18] bzip2. [Online] [Cited: 20 01 2013.]
`http://www.bzip.org`

[19] Nokia N810. [Online] [Cited: 24 01 2013.]
`http://wiki.maemo.org/Nokia_N810`

[20] Nokia N900. [Online] [Cited: 24 01 2013.]
`http://wiki.maemo.org/N900`

[21] libcurl. [Online] [Cited: 18 01 2013.]
http://curl.haxx.se/libcurl/

[22] libmicrohttpd. [Online] [Cited: 24 01 2013.]
http://www.gnu.org/software/libmicrohttpd/

[23] maemo. [Online] [Cited: 08 01 2013.]
http://maemo.org/

[24] netem. [Online] [Cited: 12 01 2013.]
http://www.linuxfoundation.org/collaborate/workgroups/networking/netem

[25] NI cRIO-9215. [Online] [Cited: 14 01 2013.]
http://www.ni.com/pdf/manuals/373779f_0118.pdf

[26] jtg. [Online] [Cited: 19 01 2013.]
http://www.netlab.tkk.fi/~jmanner/jtg.html

[27] Alexa Traffic Rank [Online] [Cited: 10 03 2013.]
http://www.alexa.com/